ITAT is a computer science conference with the primary goal of presenting new results of young researchers and doctoral students from Slovakia and Czech Republic. The conference serves as a platform for exchange of information within the community, and also provides opportunities for informal meetings of the participants in a mountainous regions of Czech Republic and Slovakia. The traditional topics of the conference include software engineering, data processing and knowledge representation, information security, theoretical foundations of computer science, computational intelligence, distributed computing, natural language processing, and computer science education. The conference accepts papers describing original previously unpublished results, significant work-in-progress reports, as well as reviews of special topics of interest to the conference audience.

The conference is organized by:

Institute of Computer Science of University of P. J. Šafárik, Košice

Institute of Computer Science of Academy of Sciences of the Czech Republic, Prague

Faculty of Mathematics and Physics, School of Computer Science, Charles University, Prague

Slovak Society for Artificial Intelligence

Stanislav Krajči (Ed.)

Information Technologies – Applications and Theory

Conference on Theory and Practice of Information Technologies
Hotel Plejsy, Krompachy, Slovakia, September 2018
Proceedings

ITAT 2018: Information Technologies – Applications and Theory

Proceedings of the 18th conference ITAT 2018

Plejsy, Slovakia, September 21–25, 2018

ITAT 2018: Information Technologies – Applications and Theory
Proceedings of the 18th conference ITAT 2018
Plejsy, Slovakia, September 21–25, 2018
Stanislav Krajči (ed.)
Cover design: Róbert Novotný
Cover photo: Róbert Novotný st.

Published online by CEUR Workshop Proceedings vol. 2203
http://ceur-ws.org/Vol-2203
ISSN 1613-0073

These proceedings contain papers from the conference ITAT 2018. All authors agreed to publish their papers in these proceedings. All papers were reviewed by at least two anonymous referees.

http://itat.ics.upjs.sk/

Preface

The 18th annual conference ITAT took place in Hotel Plejsy, Slovakia, from 21 to 25 September 2018.

ITAT, meaning Information Technologies – Applications and Theory, is a place where young researchers, doctoral students and their teachers meet every year to present their results from the various fields of computer science. They come mainly from the Czech and Slovak republic, but the conference is open to participants from other countries as well. In addition to the professional program, much time is devoted to informal discussions, and above all to the formation and maintenance of friendly relations between scientists from the both countries. The Slovak mountains make a perfect background for that. The 2018 conference consisted of the main track and 4 workshops:

- Computational Intelligence and Data Mining (CIDM)
- Slovenskočeský NLP workshop (SloNLP)
- WWW-Challenges and Trends (W3-ChaT)
- Combinatorics on words (CoW)

All the 37 submitted papers were reviewed by two or more independent reviewers. The proceedings contain 26 scientific papers and abstracts of four invited talks:

- Radim Bělohlávek (Univerzita Palackého v Olomouci): *Fuzzy Logic and Mathematics: A Historical Perspective*
- Katarína Cechlárová (Univerzita P.J.Šafárika v Košiciach): *Budúci učitelia do škôl - problém (nielen) matematický*
- Filip Šroubek (AVČR): *Digital image restoration: blur as a motion cue*
- Petr Ambrož (ČVUT): *Palindromy pohledem kombinatoriky na slovech*

The conference was organized by Institute of Computer Science, The Faculty of Natural Sciences, P. J. Šafárik University, Košice, The Academy of Sciences of the Czech Republic, Prague, The Faculty of Mathematics and Physics, Charles University, Prague, Slovenská spoločnosť pre umelú inteligenciu (Slovak society for artificial intelligence).

I would like to thank all organizers led by the conference spirit Peter Gurský, also to workshop organizers, program committee members, all anonymous referees, the invited speakers and authors of the submitted papers for contributing to the scientific program of ITAT 2018. Special thanks belong to our sponsor Profinit (http://www.profinit.eu/).

Stanislav Krajči
Šafárik University, Košice
Chair of the Program Committee

Steering Committee of ITAT 2018

Peter Vojtáš, Charles University in Prague (chair)
Martin Holeňa, Academy of Sciences of the Czech Republic
Tomáš Horváth, Pavol Jozef Šafárik University in Košice
Markéta Lopatková, ÚFAL MFF UK
Tomáš Vinař, Comenius University in Bratislava

Program Committee of ITAT 2018

David Bednárek, Charles University, Prague
Mária Bieliková, Slovak University of Technology in Bratislava
Broňa Brejová, Comenius University in Bratislava
Jiří Dokulil, University of Vienna
Jaroslava Hlaváčová, Charles University, Prague
Martin Holeňa, Czech Academy of Sciences
Tomáš Horváth, Pavol Jozef Šafárik University in Košice
Daniela Chudá, Slovak University of Technology in Bratislava
Jozef Jirásek, Pavol Jozef Šafárik University in Košice
Jana Katreniaková, Comenius University in Bratislava
Stanislav Krajči, Pavol Jozef Šafárik University in Košice (chair)
Rastislav Kráľovič, Comenius University in Bratislava
Michal Krátký, VŠB-Technical University of Ostrava
Věra Kůrková, Czech Academy of Sciences
Markéta Lopatková, Charles University, Prague
Dana Pardubská, Comenius University in Bratislava
Martin Plátek, Charles University, Prague
Karel Richta, Charles University, Prague
Gabriel Semanišin, Pavol Jozef Šafárik in Košice
Roman Špánek, Czech Academy of Sciences
Ondrej Šuch, Matej Bel University Banská Bystrica
Tomáš Vinař, Comenius University in Bratislava
Filip Zavoral, Charles University, Prague

Organizing Committee

Peter Gurský, Pavol Jozef Šafárik University in Košice (chair)
Ľubomír Antoni, Pavol Jozef Šafárik University in Košice
Alexander Szabari, Pavol Jozef Šafárik University in Košice

Contents

CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Radim Bělohlávek

Fuzzy Logic and Mathematics: A Historical Perspective

Radim Bělohlávek

Univerzita Palackého v Olomouci

Abstract: Cílem přednášky je představit stejnojmennou knihu, která nedávno vyšla (R. Bělohávek, J. W. Dauben, G. J. Klir, Fuzzy Logic and Mathematics: A Historical Perspective, Oxford University Press, New York, 2017, 544 str.). Během přednášky se podíváme na význačné myšlenkové počiny v oblasti filozofie, matematiky a přírodních věd, které předcházely vzniku fuzzy logiky, projdeme vybranými milníky vývoje teoretických základů i aplikací fuzzy logiky a zhodnotíme její dosavadní význam a perspektivy.

Radim Bělohlávek se zabývá fuzzy logikou a použitím algebry a logiky v modelování relačních dat. Je autorem nebo spoluautorem více než sta prací publikovaných v mezinárodních časopisech a čtyř knih v těchto oblastech. Působí jako profesor informatiky na Přírodovědecké fakultě Univerzity Palackého v Olomouci, kde v současné době vede katedru informatiky. Tři roky také pracoval jako řádný profesor na State University of New York v USA. Podle Web of Science byly jeho práce citovány více než 2000krát.

CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073,

Budúci učitelia do škôl – problém (nielen) matematický

Katarína Cechlárová

Univerzita P. J. Šafárika v Košiciach

Abstract: Prednáška je venovaná téme na rozhraní teoretickej informatiky a kombinatorickej optimalizácie. Predstavíme niekoľko vo svete fungujúcich priraďovacích schém, ktoré pomáhajú nájsť prvé zamestnanie absolventom medicíny, miesto v škole pre dieťa, alebo obličku pre pacienta čakajúceho na transplantáciu, a potom sa sústredíme na praktický problém, ktorý rieši Univerzita Pavla Jozefa Šafárika v Košiciach. Ide o priraďovanie študentov učiteľstva na praxe do škôl. Problém priradenia všetkých zapísaných študentov pri dodržaní kapacitných obmedzení škôl je NP-úplný, dokonca niektoré jeho verzie sú neaproximovateľné. Ukážeme, ako sme našli riešenie metódami celočíselného lineárneho programovania a ako to uľahčilo život pracovníčkam Katedry pedagogiky.

Katarína Cechlárová pôsobí na Ústave matematických vied Univerzity Pavla Jozefa Šafárika v Košiciach. Vedecky pôsobí v oblasti na rozhraní ekonómie, kombinatorickej optimalizácie a teórie výpočtovej zložitosti, nazývanej Computational Social Choice. Má širokú medzinárodnú spoluprácu. Jej spoluautormi sú kolegovia z Česka, Maďarska, Anglicka, Škótska, Grécka, Španielska, USA a Francúzska.

S. Krajči (ed.): ITAT 2018 Proceedings, pp. 3–9
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Viliam Kačala, Lukáš Miňo, and Csaba Török

Speedup of Uniform Bicubic Spline Interpolation

Viliam Kačala, Lukáš Miňo, and Csaba Török

Institute of Computer Science, Faculty of Science, Pavol Jozef Šafárik University in Košice
Jesenná 5, 040 01 Košice, Slovakia
viliam.kacala@student.upjs.sk, lukas.mino@upjs.sk, csaba.torok@upjs.sk

Abstract: The goal of the paper is to introduce an efficient algorithm for computation of derivatives of bicubic spline surfaces over equispaced grids with C^2 class continuity. The algorithm is based on a recently proposed approach using a special approximation property between quartic and cubic polynomials. The proposed solution replaces the classical de Boor's systems of equations with systems of reduced size and simple remainder explicit formulas. We will show that the proposed new algorithm is numerically equivalent to de Boor's algorithm and the former is more than 50% faster.

1 Introduction and problem statement

Modeling of surfaces plays a key role in a wide variety of computer science fields such as graphics or CAD applications. The paper proposes an efficient computational algorithm for interpolating uniform bicubic spline surfaces of class C^2. The standard way to accomplish that is an almost 60 years old classic algorithm designed by Carl de Boor [2] that uses tridiagonal systems of equations. We will refer to it as the *full* algorithm.

Even though the evaluation of such systems is quite straightforward in general, running in linear time, it can be still viewed as slow especially in real time computing scenarios. For this reason we present a new *reduced* algorithm for uniform bicubic spline surfaces that, while being in the same complexity class, is faster and needs lower memory requirements.

The structure of the paper is as follows. The remaining part of this section is devoted to the problem statement and introducing terminology. Section 2 presents the new reduced algorithm along with proof of its equality to the full algorithm. The third section provides experimental measurements of actual time savings of the new approach along with some words about its efficient implementation. To be self contained, we provide the equations of de Boor's full algorithm for surfaces and the reduced algorithm for curves in Appendix, for easier comparison with ours.

Now let's jump into the problem statement. Consider a uniform grid

$$[x_0, x_1, x_2, \ldots, x_{I-1}] \times [y_0, y_1, y_2, \ldots, y_{J-1}], \qquad (1)$$

where

$$x_i = x_0 + ih_x, \quad i = 1,2,3,\ldots,I-1, \quad I > 1, h_x \in \mathbb{R}^+,$$
$$y_j = y_0 + ih_y, \quad j = 1,2,3,\ldots,J-1, \quad J > 1, h_y \in \mathbb{R}^+.$$

According to [2], a spline surface is defined by given values

$$z_{i,j}, \quad i = 0,1,2,\ldots,I-1, \quad j = 0,1,2,\ldots,J-1 \qquad (2)$$

at the grid-points, and given first directional derivatives

$$d^x_{i,j}, \quad i = 0, I-1, \quad j = 0,1,2,\ldots,J-1 \qquad (3)$$

at the boundary verticals,

$$d^y_{i,j}, \quad i = 0,1,2,\ldots,I-1, \quad j = 0, J-1 \qquad (4)$$

at the boundary horizontals and cross derivatives

$$d^{x,y}_{i,j}, \quad i = 0, I-1, \quad j = 0, J-1 \qquad (5)$$

at the four corners of the grid.

The task is to define a quadruple $[z_{i,j}, d^x_{i,j}, d^y_{i,j}, d^{x,y}_{i,j}]$ at every grid-point $[x_i, y_j]$, based on which a bicubic clamped spline surface S of class C^2 is constructed with properties

$$S(x_i,y_j) = z_{i,j}, \qquad \frac{\partial S(x_i,y_j)}{\partial y} = d^y_{i,j},$$
$$\frac{\partial S(x_i,y_j)}{\partial x} = d^x_{i,j}, \qquad \frac{\partial^2 S(x_i,y_j)}{\partial x \partial y} = d^{x,y}_{i,j}.$$

For $I = 7$ and $J = 5$, we provide visual illustration of the input situation in Figure 1, where bold values represent (2) – (5) while the remaining non-bold values represent the unknown derivatives to compute.

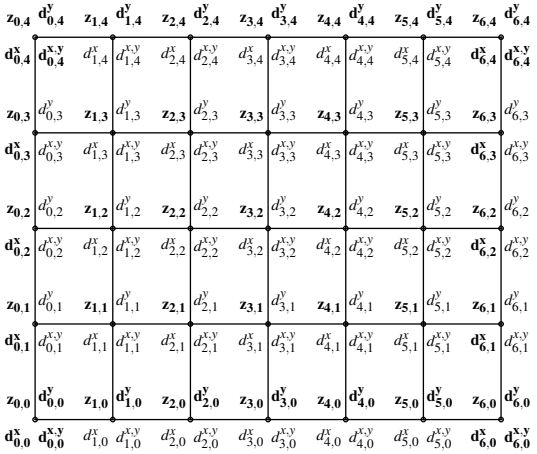


Figure 1: Input situation for a 7×5 uniform grid.

2 Reduced algorithm

The reduced algorithm for uniform cubic spline curves was proposed in [11], where the model equations were obtained using a special approximation property between cubic splines and quartic polynomials [10]. This property was extended for spline surfaces proving that a biquartic polynomial fully determines a 2×2-component uniform bicubic spline surface of class C^2 [9], [5]. This approach was generalized in [3], [6], [4], where a new algorithm for solving the unknown derivatives was proposed and new model equations and explicit formulas were derived. The proposed algorithm introduced equation systems with reduced dimensionality while the remaining unknown derivatives were computed from simple explicit formulas. Measured results showed, that the proposed algorithm is faster than the full one [4]. Thorough analysis has shown that we can further increase the performance of the reduced approach. In the next subsection every model equation will be expressed by simple expressions that correspond to the model equations of [11].

This section consists of two parts. Firstly we show that one of the four systems of de Boor's algorithm can be equivalently solved by a reduced system and additional explicit formulas. In the second part a reduced algorithm is proposed for the solution of the unknown derivatives of a spline surface of class C^2 that is equivalent to de Boor's full algorithm.

2.1 Model equations

Consider the uniform grid (1) with given input values (2) – (5). Any of the remaining unknown values can be computed using the systems of equations (20) – (23) of the full algorithm, see Appendix.

Consider a system of linear equations for the j-th horizontal ($j=0,1,2,\ldots,J-1$), see (20) in Appendix, that takes a form

$$\begin{bmatrix} 4 & 1 & 0 & & & \\ 1 & 4 & 1 & & & \\ 0 & 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & & \\ & & & 1 & 4 & 1 \\ & & & 0 & 1 & 4 \end{bmatrix} \cdot \begin{bmatrix} d^x_{1,j} \\ d^x_{2,j} \\ d^x_{3,j} \\ \vdots \\ d^x_{I-3,j} \\ d^x_{I-2,j} \end{bmatrix} = \begin{bmatrix} \frac{3}{h_x}(z_{2,j}-z_{0,j}) - d^x_{0,j} \\ \frac{3}{h_x}(z_{3,j}-z_{1,j}) \\ \frac{3}{h_x}(z_{4,j}-z_{2,j}) \\ \vdots \\ \frac{3}{h_x}(z_{I-2,j}-z_{I-4,j}) \\ \frac{3}{h_x}(z_{I-1,j}-z_{I-3,j}) - d^x_{I-1,j} \end{bmatrix}. \tag{6}$$

The following lemma shows that any of the systems (6) can be solved by a reduced system.

Lemma 1. *Let $h_x \in \mathbb{R}^+$, $I \geq 4$, $\{z_{0,j}, z_{1,j}, z_{2,j}, \ldots z_{I-1,j}, d^x_{0,j}, d^x_{I-1,j}\} \subset \mathbb{R}$ be some known values and $\{d^x_{1,j}, d^x_{2,j}, d^x_{3,j}, \ldots, d^x_{I-2,j}\} \subset \mathbb{R}$ be unknown values. Then the full equation system (6) and the reduced one*

$$\begin{bmatrix} -14 & 1 & 0 & & & \\ 1 & -14 & 1 & & & \\ 0 & 1 & -14 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -14 & 1 \\ & & & 0 & 1 & \mu \end{bmatrix} \cdot \begin{bmatrix} d^x_{2,j} \\ d^x_{4,j} \\ d^x_{6,j} \\ \vdots \\ d^x_{\nu-2,j} \\ d^x_{\nu,j} \end{bmatrix} = \begin{bmatrix} \frac{3}{h_x}(z_{4,j}-z_{0,j}) - \frac{3}{h_x}(z_{3,j}-z_{1,j}) - d^x_{0,j} \\ \frac{3}{h_x}(z_{6,j}-z_{2,j}) - \frac{3}{h_x}(z_{5,j}-z_{3,j}) \\ \frac{3}{h_x}(z_{8,j}-z_{4,j}) - \frac{3}{h_x}(z_{7,j}-z_{5,j}) \\ \vdots \\ \frac{3}{h_x}(z_{\nu,j}-z_{\nu-4,j}) - \frac{12}{h_x}(z_{\nu-1,j}-z_{\nu-3,j}) \\ \frac{3}{h_x}(z_{\nu+\tau,j}-z_{\nu-2,j}) - \frac{12}{h_x}(z_{\nu+1,j}-z_{\nu-1,j}) - \eta d_{I-1,j} \end{bmatrix}, \tag{7}$$

where

$$\begin{aligned} &\mu=-14,\ \tau=2,\ \eta=1,\ \nu=I-3, && \text{if } I \text{ is odd},\\ &\mu=-15,\ \tau=0,\ \eta=-4,\ \nu=I-2, && \text{if } I \text{ is even}, \end{aligned} \tag{8}$$

with explicit formulas

$$\begin{aligned} &d^x_{i,j} = \frac{3}{4h_x}(z_{i+1,j}-z_{i-1,j}) - \frac{1}{4}(d^x_{i+1,j}+d^x_{i-1,j}),\\ &i=1,3,5,\ldots,\nu+\tau-1, \end{aligned} \tag{9}$$

are equivalent.

Proof. It follows from Lemma 3 of Appendix that the solution of the reduced algorithm is equivalent to the solution of the full system, see e.g. [8],

$$\begin{bmatrix} 4 & 1 & 0 & & & \\ 1 & 4 & 1 & & & \\ 0 & 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & & \\ & & & 1 & 4 & 1 \\ & & & 0 & 1 & 4 \end{bmatrix} \cdot \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{N-3} \\ d_{N-2} \end{bmatrix} = \begin{bmatrix} \frac{3}{h}(y_2-y_0) - d_0 \\ \frac{3}{h}(y_3-y_1) \\ \frac{3}{h}(y_4-y_2) \\ \vdots \\ \frac{3}{h}(y_{N-2}-y_{N-4}) \\ \frac{3}{h}(y_{N-1}-y_{N-3}) - d_{N-1} \end{bmatrix}. \tag{10}$$

As we see the systems (6) and (10) use the same tridiagonal matrix. The systems (10) and (6) differ in the unknowns and the right-hand side vectors. However, using a substitution

$$h=h_x,\ d_i=d^x_{i,j},\ y_i=z_{i,j},\ i=0,1,2,\ldots,N-1,\ N=I, \tag{11}$$

we get (6) from (10), (7) from (24) and the remainder formulas (9) from (26) that alongside with Lemma 3 finishes the proof. □

2.2 Algorithm

Now we have sufficient theoretical foundations to propose an efficient algorithm to compute spline derivatives over a uniform grid of points. The reduced algorithm consists of four main steps, each evaluating systems of equations derived from (7) and remainder formulas (9). The algorithm as a whole takes the same input values and provides identical results as the original full algorithm in Appendix. Therefore it is intended as a drop-in replacement for the full algorithm.

Based on Lemma 1 the full systems (21) – (23) can be solved using analogical reduced systems and explicit formulas. Hence we get immediately the reduced algorithm for solving the unknown derivatives of a C^2 spline surface that appears to provide faster computations than de Boor's algorithm.

Theorem 1 (Reduced algorithm for surfaces)**.** *If the values (2) – (5) over the uniform grid (1) are given, then the unknown values*

$$\begin{aligned}
&d^x_{i,j}, && i=1,2,3,\ldots,I-2, && j=0,1,2,\ldots,J-1,\\
&d^y_{i,j}, && i=0,1,2,\ldots,I-1, && j=1,2,3,\ldots,J-2,\\
&d^{x,y}_{i,j}, && i=1,2,3,\ldots,I-2, && j=0,J-1,\\
&d^{x,y}_{i,j}, && i=0,1,2,\ldots,I-1, && j=1,2,3,\ldots,J-2
\end{aligned}$$

for spline surface of class C^2 are uniquely determined by the following linear systems and formulas in four main steps:

Step 1a. Computation of d^x along the horizontals from equation systems for inner even-indexed grid-points.
For each $j=0,1,2,\ldots,J-1$,

$$\begin{aligned}
&solve_system(\\
&\quad d^x_{i+2,j}-14d^x_{i,j}+d^x_{i-2,j}=\\
&\qquad=\frac{3}{h_x}(z_{i+2,j}-z_{i-2,j})-\frac{12}{h_x}(z_{i+1,j}-z_{i-1,j}), \quad (12)\\
&\quad where\ i=2,4,6,\ldots,I-3\\
&).
\end{aligned}$$

Step 1b. Computation of d^x along the horizontals from explicit formulas for inner odd-indexed grid-points.
For each $i=1,3,5,\ldots,I-2$ and $j=0,1,2,\ldots,J-1$,

$$d^x_{i,j}=\frac{3}{4h_x}(z_{i+1,j}-z_{i-1,j})-\frac{1}{4}(d^x_{i+1,j}+d^x_{i-1,j}). \quad (13)$$

Step 2a. Computation of d^y along the verticals from equation systems for inner even-indexed grid-points.
For each $i=0,1,2,\ldots,I-1$,

$$\begin{aligned}
&solve_system(\\
&\quad d^y_{i,j+2}-14d^y_{i,j}+d^y_{i,j-2}=\\
&\qquad=\frac{3}{h_y}(z_{i,j+2}-z_{i,j-2})-\frac{12}{h_y}(z_{i,j+1}-z_{i,j-1}), \quad (14)\\
&\quad where\ j=2,4,6,\ldots,J-3\\
&).
\end{aligned}$$

Step 2b. Computation of d^y along the verticals from explicit formulas for inner odd-indexed grid-points.
For each $i=0,1,2,\ldots,I-1$ and $j=1,3,5,\ldots,J-2$,

$$d^y_{i,j}=\frac{3}{4h_y}(z_{i,j+1}-z_{i,j-1})-\frac{1}{4}(d^y_{i,j+1}+d^y_{i,j-1}). \quad (15)$$

Step 3a. Computation of $d^{x,y}$ along the horizontals from equation systems for inner even-indexed grid-points.
For each $j=0,J-1$,

$$\begin{aligned}
&solve_system(\\
&\quad d^{x,y}_{i+2,j}-14d^{x,y}_{i,j}+d^{x,y}_{i-2,j}=\\
&\qquad=\frac{3}{h_x}(d^x_{i+2,j}-d^x_{i-2,j})-\frac{12}{h_x}(d^x_{i+1,j}-d^x_{i-1,j}), \quad (16)\\
&\quad where\ i=2,4,6,\ldots,I-3\\
&).
\end{aligned}$$

Step 3b. Computation of $d^{x,y}$ along the horizontals from explicit formulas for inner odd-indexed grid-points.
For each $i=1,3,5,\ldots,I-2$ and $j=0,J-1$,

$$d^{x,y}_{i,j}=\frac{3}{4h_x}(d^x_{i+1,j}-d^x_{i-1,j})-\frac{1}{4}(d^{x,y}_{i+1,j}+d^{x,y}_{i-1,j}). \quad (17)$$

Step 4a. Computation of $d^{x,y}$ along the verticals from equation systems for inner even-indexed grid-points.
For each $i=0,1,2,\ldots,I-1$,

$$\begin{aligned}
&solve_system(\\
&\quad d^{x,y}_{i,j+2}-14d^{x,y}_{i,j}+d^{x,y}_{i,j-2}=\\
&\qquad=\frac{3}{h_y}(d^y_{i,j+2}-d^y_{i,j-2})-\frac{12}{h_y}(d^y_{i,j+1}-d^y_{i,j-1}), \quad (18)\\
&\quad where\ j=2,4,6,\ldots,J-3\\
&).
\end{aligned}$$

Step 4b. Computation of $d^{x,y}$ along the verticals from explicit formulas for inner odd-indexed grid-points.
for each $i=0,1,2,\ldots,I-1$ and $i=1,3,5,\ldots,I-2$,

$$d^{x,y}_{i,j}=\frac{3}{4h_y}(d^y_{i,j+1}-d^y_{i,j-1})-\frac{1}{4}(d^{x,y}_{i,j+1}+d^{x,y}_{i,j-1}). \quad (19)$$

If I is even, then the last model equation in steps (12) and (16) needs to be accordingly replaced by the model equation derived according to (8). Analogically, if J is even, the same applies to steps (14) and (18).

We underline that the systems (12), (14), (16) and (18) are diagonally dominant and use the same tridiagonal matrix with constant main diagonal and upper, lower diagonals.

In the next section it will be shown that the reduced algorithm is faster than the full one and additionally it needs lower memory requirements.

The main difference between original full algorithm and the reduced one is that the latter computes only half of the unknown values using LU factorization where the remaining half of the unknowns are solved by simple remainder formulas. Because of this, the reduced algorithm is supposed to be more efficient in terms of execution speed and memory requirements as well. We will discuss and measure it's speed in Section 3.

3 Performance evaluation

To be worth of actual implementation, the reduced algorithm should be significantly faster than full one. The elementary task in both algorithms is computation of tridiagonal systems of equations. As the full and the reduced systems are diagonally dominant we use a modified LU factorization method as the internal solver for such systems [4]. Our optimizations of the LU factorization relies on a fact, that the left-hand side matrices comprise only constant values.

While the LU factorization for tridiagonal linear systems is a quite efficient way to solve the equations which are interdependent, it has some drawbacks in the sense of modern CPU's capabilities like usage of vectorized SIMD instructions especially due to the structure of tridiagonal matrices. By breaking down the computational task of the full approach into reduced systems and simple explicit formulas, half of the equations are independent and therefore more versatile to manual or automatic compiler optimizations.

Memory requirements For the sake of completeness some words about applied data structures and memory requirements should be given.

The input grid (1) of size $I \times J$ needs $I+J$ memory to store x and y coordinates of the total IJ grid-points. Each grid-point however requires $4IJ$ space for $z_{i,j}$, $d^x_{i,j}$, $d^y_{i,j}$, $d^{x,y}_{i,j}$ values, where most of them will be computed by the full or reduced algorithm. This gives us $4IJ+I+J$ space requirement to store input/output values.

The needs of the full and reduced algorithms are quite low regarding the size of the grid. The full tridiagonal systems in Lemma 2 require $2 \cdot max(I,J)$ space to store the right-hand side vector and an auxiliary buffer vector used for the LU factorization. In case of the reduced algorithm, only half number of the equations form the tridiagonal system, therefore they require only $max(I,J)$ space for the LU part.

Data structures Since the grid may contain tens of thousands or more grid-points, the most effective representation is the jagged array structure for each of the z, d^x, d^y, $d^{x,y}$ values. Each equation system from any of the two algorithms always depends on one row of a jagged array, therefore entire rows of the jagged structure can be effectively cached under the assumption, that the size of the row is not very large. Notice that the iterations for computing the $d^y_{i,j}$ and most of the $d^{x,y}_{i,j}$ values in both algorithms have interchanged indices compared to the iteration throughout the $d^x_{i,j}$ values. We mention that an efficient implementation needs to setup the jagged arrays in accordance with how we want to iterate the data [7].

3.1 Measurements

Now let us compare the implementations of both algorithms. We implemented a benchmark in C++ 17 compiled with MSVC 2017 using *-O2* optimization level and individual code generation for each tested CPU, i.e. CPUs with AVX2 support were running binaries compiled to AVX2 instruction set whereas older CPUs received for instance only SSE2 compiled binaries. Testing environments comprised several computers with various CPUs ranging from rather obsolete Nehalem to recent Skylake microarchitecture. All systems had 8 – 32 GB of RAM, SSD and Windows 10 installed. The tests were conducted on freshly booted PCs after 2 minutes of idle time without running any non-essential services or processes like web browsers, database engines, etc.

On all testing computers the tests were conducted on two datasets, a small one on a grid of size $I,J = 100$, and a large one, where grid dimensions were $I,J = 1000$. Both datasets comprised the grid $[x_0,x_1,\ldots,x_{I-1}] \times [y_0,y_1,\ldots,y_{J-1}]$, where $x_0 = -20$, $x_{I-1} = 20$, $y_0 = -20$, $y_{J-1} = 20$ and the values $z_{i,j}$, $d^x_{i,j}$, $d^y_{i,j}$, $d^{x,y}_{i,j}$, see (2) – (5), were computed from function $sin\sqrt{x^2+y^2}$ at each grid-point. These datasets were chosen arbitrarily and the behaviour of the algorithms was correct for any input values. The speedup values were gained averaging 50 measurements of each algorithm.

Tables 1 and 2 contain results for both datasets and consist of four columns. The first column contains the tested CPUs ordered by their release date. Columns two through four contain measured execution times in microseconds for both algorithms and their speed ratios.

To provide better comparison for more datasets, Table 3 provides measurements on more different input sizes, however for the sake of readability it contains measurements only from Intel i7 6700K as the fastest testing CPU.

Let us review the measured performance improvement of the reduced algorithm. Results of Table 1 say that the reduced algorithm is more than fifty percent faster than the full one as long as the grid dimensions remains relatively small to enable the entire rows of the grid as well as the auxiliary LU vectors to be cached.

	$I,J = 100$		
CPU	Full	Reduced	Speedup
Intel i5 M430	783	438	1.79
AMD A6 3650M	922	567	1.64
Intel i7 4790	482	250	1.93
Intel i7 6700K	355	190	1.86
AMD X4 845	648	347	1.86

Table 1: Comparison of full and reduced algorithms on small dataset. Times are in microseconds.

	$I,J = 1000$		
CPU	Full	Reduced	Speedup
Intel i5 M430	116 856	90 541	1.29
AMD A6 3650M	116 390	78 892	1.48
Intel i7 4790	54 719	33 584	1.63
Intel i7 6700K	37 029	22 571	1.64
AMD X4 845	78 860	48 955	1.61

Table 2: Comparison of full and reduced algorithms on large dataset. Times are in microseconds.

The speedup of the reduced algorithm in Table 2 with a larger dataset is not so high due to difficulties of keeping data in fast but small L1 cache, resulting in higher ratio of cache misses. In the case of even larger dataset, let's say in the order of billions of grid-points, the performances of both algorithms are similar with the reduced one gaining only small advantage.

In our experiments we observed that the reduced algorithm was always faster than the full one, however exact speedup depends on the size of the input grid. The general rule is: the larger the grid the smaller the speedup.

The highest speedup does not depend on the grid size if it is from range of 50 to 200.

CPU	Full	Reduced	Speedup
$I,J = 50$	92	50	1.84
$I,J = 100$	355	190	1.86
$I,J = 200$	1 417	778	1.82
$I,J = 300$	3 203	1 797	1.78
$I,J = 400$	5 791	3 234	1.79
$I,J = 1000$	37 029	22 571	1.64
$I,J = 1500$	88 236	54 141	1.63
$I,J = 2000$	178 144	115 674	1.54

Table 3: Multiple dataset comparison of full and reduced algorithms tested on i7 6700K. Times are in microseconds.

4 Discussion

Let us briefly discuss the new algorithm from the numerical and experimental point of view. While the classic full algorithm is composed of four series of tridiagonal systems of equations, the reduced algorithm breaks down each equation system into a reduced one approximately half the size of the original one and simple mutually independent explicit formulas.

In addition, from the numerical point of view, the reduced tridiagonal subsystems are diagonally dominant and therefore computationally stable [1], similarly to the full systems. The remainder explicit formulas (9) are simple and thus do not present an issue.

The maximal numerical difference in our C++ implementation was in the order of 10^{-16} on several different datasets so the reduced algorithm yields numerically accurate results.

Since the algorithm consists of many independent systems of linear equations, it can be also effectively parallelized for both CPU and GPU architectures.

There is also a future work for further reduction of the number of equations in the tridiagonal systems.

5 Conclusion

The paper introduced a new algorithm to compute the unknown derivatives for uniform bicubic spline surfaces of class C^2. The algorithm reduces the size of the tridiagonal systems of equations by half and computes the remaining unknown derivatives using simple explicit formulas. A substantial decrease of execution time of derivatives at grid-points has been achieved with lower memory space requirements at the cost of a slightly more complex implementation where the measured speedup ranges from 1.3 to 1.9 depending on the grid size and CPU architecture.

Acknowledgements

This work was partially supported by projects Technicom ITMS 26220220182 and APVV-15-0091 Effective algorithms, automata and data structures.

Appendix

Carl de Boor in [2] proposed an algorithm for computation of the unknown derivatives of spline surface of class C^2 over any grid with four types of full systems of linear equations. The following lemma reformulates this algorithm for uniform grid.

Lemma 2 (Full algorithm for surfaces)**.** *If the values (2) – (5) over the uniform grid (1) are given, then the unknown*

values

$$\begin{aligned} &d^x_{i,j}, && i=1,2,3,\ldots,I-2, && j=0,1,2,\ldots,J-1,\\ &d^y_{i,j}, && i=0,1,2,\ldots,I-1, && j=1,2,3,\ldots,J-2,\\ &d^{x,y}_{i,j}, && i=1,2,3,\ldots,I-2, && j=0,J-1,\\ &d^{x,y}_{i,j}, && i=0,1,2,\ldots,I-1, && j=1,2,3,\ldots,J-2 \end{aligned}$$

for spline surface of class C^2 are uniquely determined by the following linear systems of equations:

Step 1. Computation of d^x along the horizontals from equation systems.

For each $j=0,1,2,\ldots,J-1$,

solve_system(

$$d^x_{i+1,j}+4\cdot d^x_{i,j}+d^x_{i-1,j}=\frac{3}{h_x}\cdot(z_{i+1,j}-z_{i-1,j}), \quad \text{where } i=1,2,3,\ldots,I-2 \tag{20}$$

).

Step 2. Computation of d^y along the verticals from equation systems.

For each $i=0,1,2,\ldots,I-1$,

solve_system(

$$d^y_{i,j+1}+4\cdot d^y_{i,j}+d^x_{i,j-1}=\frac{3}{h_y}\cdot(z_{i,j+1}-z_{i,j-1}), \quad \text{where } j=1,2,3,\ldots,J-2 \tag{21}$$

).

Step 3. Computation of $d^{x,y}$ along the horizontals from equation systems.

For each $j=0,J-1$,

solve_system(

$$d^{x,y}_{i+1,j}+4\cdot d^{x,y}_{i,j}+d^{x,y}_{i-1,j}=\frac{3}{h_x}\cdot(d^x_{i+1,j}-d^x_{i-1,j}), \quad \text{where } i=1,2,3,\ldots,I-2 \tag{22}$$

).

Step 4. Computation of $d^{x,y}$ along the verticals from equation systems.

For each $i=0,1,2,\ldots,I-1$,

solve_system(

$$d^{z,y}_{i,j+1}+4\cdot d^{z,y}_{i,j}+d^{z,y}_{i,j-1}=\frac{3}{h_y}\cdot(d^y_{i,j+1}-d^y_{i,j-1}), \quad \text{where } j=1,2,3,\ldots,J-2 \tag{23}$$

).

A new algorithm was proposed in [11] for computation of the unknown derivatives of a spline curve of class C^2 over uniform grids with a reduced system of linear equations instead of the well known full algorithm with parameters 1,4,1.

Lemma 3 (Reduced algorithm for curves)**.** *Consider a cubic clamped spline of class C^2 over a uniform grid $[u_0,u_1,\ldots,u_{N-1}]$, where $u_i=u_0+ih$, $i=1,2,3,\ldots,N-1$, defined by given values $y_0,\ldots,y_{N-1}$ and d_0, d_{N-1} at grid-points. The tridiagonal system*

$$\begin{bmatrix} -14 & 1 & 0 & & & & \\ 1 & -14 & 1 & & & & \\ 0 & 1 & -14 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & 1 & -14 & 1 & \\ & & & 0 & 1 & -14 \end{bmatrix} \cdot \begin{bmatrix} d_2 \\ d_4 \\ d_6 \\ \vdots \\ d_{\nu-2} \\ d_\nu \end{bmatrix} =$$

$$= \begin{bmatrix} \frac{3}{h}(y_4-y_0)-\frac{12}{h}(y_3-y_1)-d_0 \\ \frac{3}{h}(y_6-y_2)-\frac{12}{h}(y_5-y_3) \\ \frac{3}{h}(y_8-y_4)-\frac{12}{h}(y_7-y_5) \\ \vdots \\ \frac{3}{h}(y_\nu-y_{\nu-4})-\frac{12}{h}(y_{\nu-1}-y_{\nu-3}) \\ \frac{3}{h}(y_{\nu+\tau}-y_{\nu-2})-\frac{12}{h}(y_{\nu+1}-y_{\nu-1})-\eta d_{N-1} \end{bmatrix}, \tag{24}$$

where

$$\begin{aligned} &\mu=-14,\ \tau=2,\ \eta=1,\ \nu=N-3, && \text{if } N \text{ is odd},\\ &\mu=-15,\ \tau=0,\ \eta=-4,\ \nu=N-2, && \text{if } N \text{ is even}, \end{aligned} \tag{25}$$

and the formula

$$\begin{aligned} &d_i=\frac{3}{4h}(y_{i+1}-y_{i-1})-\frac{1}{4}(d_{i+1}+d_{i-1}),\\ &i=1,3,5,\ldots,\nu+\tau-1, \end{aligned} \tag{26}$$

imply that the second derivatives of spline components at the inner grid-points are equal.

References

[1] Björck A: Numerical Methods in Matrix Computations. Springer (2015)

[2] de Boor, C.: Bicubic Spline Interpolation. Journal of Mathematics and Physics, 41(3), (1962), 212–218

[3] Miňo L.: Efficient Computational Algorithm for Spline Surfaces. ITAT 2015: Information Technologies – Applications and Theory, ISSN 1613–0073, (2015), 30–37

[4] Kačala, V., Miňo, L.: Speeding up the Computation of Uniform Bicubic Spline Surfaces. WSCG 2017: 25. Conference on Computer Graphics, Visualization and Computer Vision 2017, ISSN 2464–4617, (2017), 73–80

[5] Miňo L., Szabó I., Török Cs.: Bicubic Splines and Biquartic Polynomials. Open Computer Science, Volume 6, Issue 1, ISSN (Online) 2299–1093, (2016), 1–7

[6] Miňo L., Török Cs.: Fast Algorithm for Spline Surfaces. Communication of the Joint Institute for Nuclear Research, Dubna, Russian Federation, E11-2015-77, (2015), 1–19

[7] Patterson, J. R. C.: Modern Microprocessors – A 90-Minute Guide!, Lighterra (2015)

[8] Salomon, D.: Curves and Surfaces for Computer Graphics. Springer (2006)

[9] Szabó I.: Approximation Algorithms for 3D Data Analysis. PhD Thesis, P. J. Šafárik University in Košice, Slovakia (2016)

[10] Török Cs.: On reduction of equations' number for cubic splines. Matematicheskoe modelirovanie, Vol. 26 , No. 11, ISSN 0234–0879, (2014), 33–36

[11] Török, Cs.: Speed-up of Interpolating Spline Construction, (to appear)

S. Krajči (ed.): ITAT 2018 Proceedings, pp. 10–17
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Martin Plátek, Dana Pardubská, and František Mráz

On (In)Sensitivity by Two-Way Restarting Automata

Martin Plátek[1], Dana Pardubská[2]*, and František Mráz[1]

[1] Charles University, Department of Computer Science
Malostranské nám. 25, 118 00 PRAHA 1, Czech Republic
martin.platek@mff.cuni.cz, frantisek.mraz@mff.cuni.cz

[2] Comenius University in Bratislava, Department of Computer Science
Mlynská Dolina, 84248 Bratislava, Slovakia
pardubska@dcs.fmph.uniba.sk

Abstract: We study *h-lexicalized two-way restarting automaton* (hRLWW(i)) that can rewrite at most i times per cycle, for $i \geq 1$. This model is useful for the study of lexical syntactic disambiguation (a notion from linguistic) through the formal notion of *h-lexicalized syntactic analysis (hLSA)*. The hLSA is composed of a *basic language* and the corresponding *h-proper language* obtained from the basic language by mapping all non-input symbols on input symbols. We compare the properties of input languages, which are the languages traditionally considered in automata theory, to the properties of hLSA, i.e., to the properties of basic and h-proper languages.

The basic and h-proper languages of hRLWW(i)-automata fulfill the so called *reduction correctness preserving property*, but the input languages do not. While basic and h-proper languages are sensitive to the size of the read/write window, the input languages are not. Moreover, the basic and h-proper languages are sensitive to the number of rewrite steps per cycle. All that concerns a subclass of context-sensitive languages containing all context-free languages (and most probably also the class of mildly context-sensitive languages [5]), i.e., a class suitable for studying and classifying syntactic and semantic features of natural languages.

We work here also with the parametrized constraint of monotonicity. While using the monotonicity of degree one we can characterize the class of context-free languages, the monotonicity of higher degrees can model more complex syntactic phenomena of whole natural languages (like cross-serial dependencies [5]).

Finally, we stress the constraint of *weak cyclic form*. It preserves the power of hRLWW(i)-automata, and it allows to extend the complexity results obtained for the classes of infinite languages also into the classes of finite languages (parametrized by the number of performed cycles). It is useful for classifications in computational and corpus linguistics, where all the (syntactic) observation are of the finite nature. The main technical novelty of the paper are the results about the sensitivity and insensivity of finite and infinite hLSA and corresponding languages by hRLWW(i)-automata.

*The research is partially supported by VEGA 2/0165/16

1 Introduction

This paper is a continuation of conference papers [13], [14], and the technical report [15]. Its motivation is to study lexical syntactic disambiguation, which is one of the basic concepts of several linguistic schools, including the schools working with dependency syntax. In order to give a theoretical basis for lexicalized syntax, a model of a restarting automaton that formalizes lexicalization in a similar way as categorial grammars (see, e.g., [1]) – the *h-lexicalized restarting automaton* (hRLWW), was introduced in [13]. This model is obtained from the two-way restarting automaton of [12] by adding a letter-to-letter morphism h that assigns an input symbol to each working symbol. Then the *basic language* $L_{\mathrm{C}}(M)$ of an hRLWW-automaton M consists of all words over the working alphabet of M that are accepted by M, and the *h-proper language* $L_{\mathrm{hP}}(M)$ of M is obtained from $L_{\mathrm{C}}(M)$ through the morphism h.

Further, the set of pairs $\{\,(h(w), w) \mid w \in L_{\mathrm{C}}(M)\,\}$, denoted as $L_{\mathrm{A}}(M)$, is called the h-lexicalized syntactic analysis (hLSA) by M. Thus, in this setting the auxiliary symbols themselves play the role of the tagged items. That is, each auxiliary symbol b can be seen as a pair consisting of an input symbol $h(b)$ and some additional syntactico-semantic information (tags).

In contrast to the original hRLWW-automaton that uses exactly one rewrite in a cycle, here we study *h-lexicalized restarting automaton* (hRLWW(i)) allowing $i \geq 1$ rewrites in a cycle. Our effort is to show that this model is suited for a transparent and sensitive modeling of the lexicalized syntactic analysis (lexical disambiguation) of natural languages (compare to [8, 9]).

The lexicalized syntactic analysis based on analysis by reduction is traditionally used to analyze sentences of natural languages with a high degree of word-order freedom like, e.g., Czech, Latin, or German (see, e.g., [8]). Usually, a human reader is supposed to understand the meaning of a given sentence before he starts to analyze it; h-lexicalized syntactic analysis based on the analysis by reduction simulates such a behavior by analysis of sentences, where morphological and syntactical tags have been added to the word-forms and punctuation marks (see, e.g., [9]). We recall some constraints that are typical for restarting automata, and we outline ways for new combinations of con-

straints. We establish infinite (two-dimensional) ascending hierarchies of language classes of h-proper languages for several subtypes of hRLWW(i)-automata with respect to the number of allowed cycles, the number of symbols deleted during each cycle or the length of scanning window of an automaton.

The paper is structured as follows. In Section 2, we introduce our model and its sub-models, we define the h-proper languages, and we state the reduction correctness preserving property and the reduction error preserving property for the basic languages of deterministic h-RLWW-automata. In Section 3, we present the achieved results. The paper concludes with Section 4 in which we summarize our results and state some problems for future work.

2 Definitions

By $\subseteq$ and $\subset$ we denote the subset and the proper subset relation, respectively. Further, we will sometimes use regular expressions instead of the corresponding regular languages. Finally, throughout the paper λ will denote the empty word.

We start with the definition of the two-way restarting automaton as an extension to the original definition from [12]. In contrast to [14], we do not introduce general h-lexicalized two-way restarting list automaton which can rewrite arbitrary many times during each cycle. Instead, we introduce two-way restarting automata which can rewrite at most i times per cycle, for an integer $i \geq 1$.

Definition 1. *Let i be a positive integer. A* two-way restarting automaton*, an* RLWW*(i)-automaton for short, is a machine with a single flexible tape and a finite-state control. It is defined through an 9-tuple $M = (Q, \Sigma, \Gamma, \mathfrak{c}, \$, q_0, k, i, \delta)$, where Q is a finite set of states, Σ is a finite input alphabet, and $\Gamma (\supseteq \Sigma)$ is a finite working alphabet. The symbols from $\Gamma \setminus \Sigma$ are called* auxiliary symbols*. Further, the symbols $\mathfrak{c}, \$ \notin \Gamma$, called* sentinels*, are the markers for the left and right border of the workspace, respectively, $q_0 \in Q$ is the initial state, $k \geq 1$ is the size of the* read/write window*, $i \geq 1$ is the number of allowed rewrites in a cycle (see later), and*

$$\delta : Q \times \mathscr{PC}^{\leq k} \to \mathscr{P}((Q \times \{\mathsf{MVR}, \mathsf{MVL}, \mathsf{W}(y), \mathsf{SL}(v)\}) \cup \{\mathsf{Restart}, \mathsf{Accept}, \mathsf{Reject}\})$$

is the transition relation*. Here $\mathscr{P}(S)$ denotes the powerset of a set S,*

$$\mathscr{PC}^{\leq k} = (\mathfrak{c} \cdot \Gamma^{k-1}) \cup \Gamma^k \cup (\Gamma^{\leq k-1} \cdot \$) \cup (\mathfrak{c} \cdot \Gamma^{\leq k-2} \cdot \$)$$

is the set of possible contents *of the read/write window of M, $v \in \mathscr{PC}^{\leq k-1}$, and $y \in \mathscr{PC}^{\leq k}$.*

Being in a state $q \in Q$ and seeing $u \in \mathscr{PC}^{\leq k}$ in its window, the automaton can perform seven different types of transition steps (or instructions):

1. *A* move-right step $(q,u) \longrightarrow (q', \mathsf{MVR})$ *assumes that $(q', \mathsf{MVR}) \in \delta(q,u)$, where $q' \in Q$ and $u \notin \{\lambda, \mathfrak{c}\} \cdot \Gamma^{\leq k-1} \cdot \$$. This move-right step causes M to shift the window one position to the right and to enter state q'.*

2. *A* move-left step $(q,u) \longrightarrow (q', \mathsf{MVL})$ *assumes that $(q', \mathsf{MVL}) \in \delta(q,u)$, where $q' \in Q$ and $u \notin \mathfrak{c} \cdot \Gamma^* \cdot \{\lambda, \$\}$. It causes M to shift the window one position to the left and to enter state q'.*

3. *An* SL-step $(q,u) \longrightarrow (q', \mathsf{SL}(v))$ *assumes that $(q', \mathsf{SL}(v)) \in \delta(q,u)$, where $q' \in Q$, $v \in \mathscr{PC}^{\leq k-1}$, v is shorter than u, and v contains all the sentinels that occur in u (if any). It causes M to replace u by v, to enter state q', and to shift the window by $|u| - |v|$ items to the left – but at most to the left sentinel $\mathfrak{c}$ (that is, the contents of the window is 'completed' from the left, and so the distance to the left sentinel decreases, if the window was not already at $\mathfrak{c}$).*

4. *A* W-step $(q,u) \longrightarrow (q', \mathsf{W}(v))$ *assumes that $(q', \mathsf{W}(v)) \in \delta(q,u)$, where $q' \in Q$, $v \in \mathscr{PC}^{\leq k}$, $|v| = |u|$, and that v contains all the sentinels that occur in u (if any). It causes M to replace u by v, and to enter state q' without moving its window.*

5. *A* restart step $(q,u) \longrightarrow \mathsf{Restart}$ *assumes that* $\mathsf{Restart} \in \delta(q,u)$*. It causes M to place its window at the left end of its tape, so that the first symbol it sees is the left sentinel $\mathfrak{c}$, and to reenter the initial state q_0.*

6. *An* accept step $(q,u) \longrightarrow \mathsf{Accept}$ *assumes that* $\mathsf{Accept} \in \delta(q,u)$*. It causes M to halt and accept.*

7. *A* reject step $(q,u) \longrightarrow \mathsf{Reject}$ *assumes that* $\mathsf{Reject} \in \delta(q,u)$*. It causes M to halt and reject.*

A *configuration* of an RLWW(i)-automaton M is a word $\alpha q \beta$, where $q \in Q$, and either $\alpha = \lambda$ and $\beta \in \{\mathfrak{c}\} \cdot \Gamma^* \cdot \{\$\}$ or $\alpha \in \{\mathfrak{c}\} \cdot \Gamma^*$ and $\beta \in \Gamma^* \cdot \{\$\}$; here q represents the current state, $\alpha\beta$ is the current contents of the tape, and it is understood that the read/write window contains the first k symbols of β or all of β if $|\beta| < k$. A *restarting configuration* is of the form $q_0 \mathfrak{c} w \$$, where $w \in \Gamma^*$; if $w \in \Sigma^*$, then $q_0 \mathfrak{c} w \$$ is an *initial configuration*. We see that any initial configuration is also a restarting configuration, and that any restart transfers M into a restarting configuration.

In general, an RLWW(i)-automaton M is *nondeterministic*, that is, there can be two or more steps (instructions) with the same left-hand side (q,u), and thus, there can be more than one computation that start from a given restarting configuration. If this is not the case, the automaton is *deterministic*.

A *computation* of M is a sequence $C = C_0, C_1, \ldots, C_j$ of configurations of M, where C_0 is an initial or a restarting configuration and C_{i+1} is obtained from C_i by a step of M, for all $0 \leq i < j$. In the following we only consider computations of RLWW(i)-automata which are finite and end either by an accept or by a reject step.

Cycles and tails: Any finite computation of an RLWW(i)-automaton M consists of certain phases. A phase, called a *cycle*, starts in a restarting configuration, the window moves along the tape performing non-restarting steps until a restart step is performed and thus a new restarting configuration is reached. If no further restart step is performed, any finite computation necessarily finishes in a halting configuration – such a phase is called a *tail*. It is required that in each cycle RLWW(i)-automaton executes at most i rewrite steps (of type W or SL) but at least one SL-step. Moreover, it must not execute any rewrite step in a tail.

This induces the following relation of *cycle-rewriting* by M: $u \Rightarrow_M^c v$ iff there is a cycle that begins with the restarting configuration $q_0 \¢ u \$$ and ends with the restarting configuration $q_0 \¢ v \$$. The relation $\Rightarrow_M^{c^*}$ is the reflexive and transitive closure of $\Rightarrow_M^c$. We stress that the cycle-rewriting is a very important feature of an RLWW(i)-automaton. As each SL-step is strictly length-reducing, we see that $u \Rightarrow_M^c v$ implies that $|u| > |v|$. Accordingly, $u \Rightarrow_M^c v$ is also called a *reduction* by M.

An *input word* $w \in \Sigma^*$ *is accepted by* M, if there is a computation which starts with the initial configuration $q_0 \¢ w \$$ and ends by executing an accept step. By $L(M)$ we denote the language consisting of all input words accepted by M; we say that M *recognizes (or accepts) the input language* $L(M)$.

A *basic (or characteristic) word* $w \in \Gamma^*$ is accepted by M if there is a computation which starts with the restarting configuration $q_0 \¢ w \$$ and ends by executing an accept step. By $L_C(M)$ we denote the set of all words from Γ^* that are accepted by M; we say that M *recognizes (or accepts) the basic (or characteristic*[1]*) language* L_C.

Finally, we come to the definition of the central notion of this paper, the h-lexicalized RLWW(i)-automaton.

Definition 2. *Let i be a positive integer. An* h-lexicalized RLWW(i)-automaton*, or an* hRLWW(i)-automaton*, is a pair $\widehat{M} = (M,h)$, where $M = (Q,\Sigma,\Gamma,\¢,\$,q_0,k,i,\delta)$ is an* RLWW(i)*-automaton and $h : \Gamma \to \Sigma$ is a letter-to-letter morphism satisfying $h(a) = a$ for all input letters $a \in \Sigma$. The input language $L(\widehat{M})$ of $\widehat{M}$ is simply the language $L(M)$ and the basic language $L_C(\widehat{M})$ of $\widehat{M}$ is the language $L_C(M)$. Finally, we take $L_{hP}(\widehat{M}) = h(L_C(M))$, and we say that $\widehat{M}$* recognizes (or accepts) the h-proper language *$L_{hP}(\widehat{M})$.*

Finally, the set $L_A(\widehat{M}) = \{ (h(w),w) \mid w \in L_C(M) \}$ is called the h-lexicalized syntactic analysis (shortly hLSA) by $\widehat{M}$.

We say that for $x \in \Sigma^*$, $L_A(\widehat{M},x) = \{ (x,y) \mid y \in L_C(M), h(y) = x \}$ is the h-syntactic analysis (lexicalized syntactic analysis) *for x by $\widehat{M}$. We see that $L_A(\widehat{M},x)$ is non-empty only for x from $L_{hP}(\widehat{M})$.*

[1] Basic languages were called characteristic languages in [10] and several other papers, therefore, here we preserve the original notation with subscript C.

Evidently, for an hRLWW(i)-automaton $\widehat{M}$, we have $L(\widehat{M}) \subseteq L_{hP}(\widehat{M}) = h(L_C(\widehat{M}))$.

Let us note that h-lexicalized syntactic analysis formalizes the linguistic notion of *lexical disambiguation*. Each auxiliary symbol $x \in \Gamma \setminus \Sigma$ of a word from $L_C(\widehat{M})$ can be considered as a disambiguated input symbol $h(x)$.

The following two facts ensure the transparency for computations of hRLWW(i)-automata with respect to their basic and h-proper languages.

Fact 1. (Reduction Error Preserving Property) *Let M be an* hRLWW*(i)-automaton. If $u \Rightarrow_M^{c^*} v$ and $u \notin L_C(M)$, then $v \notin L_C(M)$.*

Fact 2. (Reduction Correctness Preserving Property) *Let M be a deterministic* hRLWW*(i)-automaton. If $u \Rightarrow_M^{c^*} v$ and $u \in L_C(M)$, then $v \in L_C(M)$, and $h(v) \in L_{hP}(M)$.*

Notations. For brevity, the prefix det- will be used to denote the property of being deterministic. For any class A of automata, $\mathscr{L}(\mathsf{A})$ will denote the class of input languages that are recognized by automata from A, $\mathscr{L}_C(\mathsf{A})$ will denote the class of basic languages that are recognized by automata from A, and $\mathscr{L}_{hP}(\mathsf{A})$ will denote the class of h-proper languages that are recognized by automata from A. $\mathscr{L}_A(\mathsf{A})$ will denote the class of hLSA (h-lexicalized syntactic analyses) that are defined by automata from A. For a natural number $k \geq 1$, $\mathscr{L}(k\text{-}\mathsf{A})$, $\mathscr{L}_C(k\text{-}\mathsf{A})$, $\mathscr{L}_{hP}(k\text{-}\mathsf{A})$, and $\mathscr{L}_A(k\text{-}\mathsf{A})$ will denote the class of input, basic, h-proper languages, and hLSA's, respectively, that are recognized by those automata from A that use a read/write window of size k.

2.1 Further Refinements and Constraints on RLWW-Automata (hRLWW-Automata)

Here we introduce some constrained types of rewrite steps whose introduction is motivated by different type of linguistic reductions.

A *delete-left step* $(q,u) \to (q', \mathsf{DL}(v))$ is a special type of an SL-step $(q', \mathsf{SL}(v)) \in \delta(q,u)$, where v is a proper (scattered) subsequence of u, containing all the sentinels from u (if any). It causes M to replace u by v (by deleting excessive symbols), to enter state q', and to shift the window by $|u| - |v|$ symbols to the left, but at most to the left sentinel $\¢$.

A *contextual-left step* $(q,u) \to (q', \mathsf{CL}(v))$ is a special type of DL-step $(q', \mathsf{DL}(v)) \in \delta(q,u)$, where $u = v_1u_1v_2u_2v_3$ and $v = v_1v_2v_3$ such that v contains all the sentinels from u (if any). It causes M to replace u by v (by deleting the factors u_1 and u_2 of u), to enter state q', and to shift the window by $|u| - |v|$ symbols to the left, but at most to the left sentinel $\¢$.

An RLWW(i)-automaton is called RLW(i)-automaton if its working alphabet coincides with its input alphabet, that is, no auxiliary symbols are available for this automaton. Note that in this situation, each restarting configuration is

necessarily an initial configuration. Within an abbreviation for an automata type, R denotes the use of moves to the right, L denotes the use of moves to the left, WW denotes the use of both input and working alphabets, and single W denotes the use of input alphabet only (the working alphabet coincides with the input alphabet).

An RLW(i)-automaton is called RLWD*(i)-automaton* if all its rewrite steps are DL steps, and it is an RLWC*(i)*-automaton if all its rewrite steps are CL-steps. Further, an RLWW(i)-automaton is called RLWWC*(i)-automaton* if all its rewrite steps are CL-steps. Similarly, an RLWW(i)-automaton is called RLWWD*(i)-automaton* if all its rewrite steps are DL-steps. Observe that when concentrating on input languages, DL- and CL-steps ensure that no auxiliary symbols can ever occur on the tape; if, however, we are interested in basic or h-proper languages, then auxiliary symbols can play an important role even though a given RLWW(i)-automaton uses only DL- or CL-steps. Therefore, we distinguish between RLWWC(i)- and RLWC(i)-automata, and between RLWWD(i)- and RLWD(i)-automata.

In the following we will use the corresponding notations also for subclasses of RLWW(i)-automata, and hRLWW(i)-automata.

Evidently, we need not distinguish between hRLW(i)-automata and RLW(i)-automata, since for the RLW(i)-automata the only possible morphism h is the identity.

Fact 3. (Equality of Languages for RLW(i)-automata.) *For any RLW(i)-automaton M, $L(M) = L_{\mathrm{C}}(M) = L_{\mathrm{hP}}(M)$.*

We recall the notion of *monotonicity* (see [3, 4]) as an important constraint for computations of RLWW(i)-automata. Let M be an RLWW(i)-automaton, and let $C = C_k, C_{k+1}, \ldots, C_j$ be a sequence of configurations of M, where $C_{\ell+1}$ is obtained by a single transition step from C_ℓ, $k \leq \ell < j$. We say that C is a *subcomputation* of M. If $C_\ell = \mathrm{¢}\alpha q \beta \$$, then $|\beta\$|$ is the *right distance* of C_ℓ, which is denoted by $D_{\mathrm{r}}(C_\ell)$. We say that a sub-sequence $(C_{\ell_1}, C_{\ell_2}, \ldots, C_{\ell_n})$ of C is *monotone* if $D_{\mathrm{r}}(C_{\ell_1}) \geq D_{\mathrm{r}}(C_{\ell_2}) \geq \cdots \geq D_{\mathrm{r}}(C_{\ell_n})$. A computation of M is called *monotone* if the corresponding sub-sequence of rewrite configurations is monotone. Here a configuration is called a *rewrite configuration* if in this configuration an SL- or W-step is being applied. Finally, M itself is called *monotone* if each of its computations is monotone. We use the prefix mon- to denote monotone types of RLWW(i)-automata. This notion of monotonicity has been considered before in various papers (see [7]) similarly as its following generalization.

Naturally, a mon-RLWW(i)-automaton can be used to simulate bottom-up one-pass parsers. In order to simulate also bottom-up multi-pass parsers, a j-monotonicity for restarting automata was introduced in [12]. For an integer $j \geq 1$, a RLWW(i)- automaton is called j-monotone if, for each of its computations, the corresponding sequence of rewriting configurations can be partitioned into at most j sub-sequences such that each of these sub-sequences is monotone. We use the prefix mon(j)- to denote j-monotone types of RLWW(i)-automata.

Here we transfer some restricted form of restarting automata called *weak cyclic form* (wcf) (see [3]) over to RLWW(i)'s. An RLWW M is said to be in *weak cyclic form* if $|uv| \leq k$ for each accepting configuration $\mathrm{¢}uqv\$$ of M, where k is the size of the read/write window of M. Thus, before M can accept, it must erase sufficiently many letters from its tape. The prefix wcf- will be used to denote restarting automata in the weak cyclic form.

3 Results

3.1 On the Insensitivity of Input Languages

The following characterizations, which are slight extensions of known results, show that with respect to their input languages, (monotone) RLWW(1)-automata are insensitive to the size of their read/write windows. In the following CFL is the class of *context-free languages*.

Theorem 4. *For all $k \geq 3$, the following equalities hold:*

(a) $\mathsf{CFL} = \mathscr{L}(\text{mon-RLWW}(1))$,
(b) $\mathsf{CFL} = \mathscr{L}(k\text{-wcf-mon-RLWW}(1))$,
(c) $\mathscr{L}(\text{RLWW}(1)) = \mathscr{L}(k\text{-wcf-RLWW}(1))$.

Proof. It follows from [7] that $\mathsf{CFL} = \mathscr{L}(k\text{-mon-RLWW}(1))$ for all $k \geq 3$, and the equality $\mathscr{L}(\text{RLWW}) = \mathscr{L}(k\text{-RLWW}(1))$ follows from [16].

It remains to prove that each (monotone) RLWW(1)-automaton with a window of size three can be converted into weak cyclic form.

Transformation into weak cyclic form for input languages. So let M_1 be an RLWW(1)-automaton with window size three. A wcf-RLWW(1)-automaton M_2 can simulate M_1 as follows: M_2 uses a new non-input symbol # to indicate that M_1 halts with accepting. At the beginning of each cycle M_2 checks the symbol in front of the right sentinel \$. If it is # there, M_2 simply deletes the symbol in front of # and restarts, except when # is the only symbol on the tape in which case it accepts. If there is no # on the tape, M_2 simulates M_1 step by step until M_1 is about to halt. If M_1 rejects, then so does M_2. If, however, M_1 accepts, then M_2 either accepts if the tape contents is of length at most three, or, in case of longer tape contents, instead of accepting, M_2 moves its window to the right, rewrites the last two symbols in front of the right sentinel \$ by the special symbol # and restarts.

It is easily seen that M_2 is monotone, if M_1 is, that it has window size three, and that it accepts the same input language as M_1. □

Theorem 5. *For all $k, k', j \geq 1$, $X \in \{\lambda, wcf\}$:*

(a) $\mathscr{L}(k\text{-X-RLWW}(j)) \subseteq \mathscr{L}(k'\text{-X-RLWW}(j'))$, *for all* $j' \geq \lceil \frac{k}{k'} \rceil \cdot j$.

(b) $\mathsf{CFL} \subset \mathscr{L}(2\text{-X-RLWW}(j))$.

Proof. Case (a) follows from an easy observation that a single rewrite operation using a window of size k can be simulated by at most $\lceil \frac{k}{k'} \rceil$ rewrite operations with a window of size k'. To prove (b) we show that 2-wcf-RLWW(1)-automata accept all context-free languages and give a non-context-free language accepted by 2-wcf-RLWW(1)-automaton.

Given any context-free language L, Niemann and Otto in [11] showed how to construct a monotone one-way restarting automaton with window size 3 recognizing L. As the constructed automaton rewrites at most 2 consecutive symbols in a cycle and RLWW(1)-automaton can move in both directions, the constructed automaton can be simulated by a monotone 2-RLWW(1)-automaton. This implies $\mathsf{CFL} \subseteq \mathscr{L}(\text{2-X-RLWW}(j))$. To complete the proof of (b) it remains to show that the inclusion is proper. We will show that the non-context-free language $L = \{a^n b^{2n} c^n \mid n \geq 0\}$ can be accepted by a 2-RLWW(1)-automaton M.

The automaton M will use two auxiliary symbols X, Y whose role is to indicate deleted subwords ab and bc, respectively. Within four cycles the automaton deletes one occurrence of a and c and two symbols b. To do so M scans the contents ω of the tape within the sentinels from left to right and with the right sentinel in its window M distinguishes six situations:

1. if $\omega = \lambda$ then M accepts;
2. if $\omega = a^+b^+c^+$, then M rewrites ab with X and restarts;
3. if $\omega = a^*Xb^+c^+$, then M rewrites bc with Y and restarts;
4. if $\omega = a^*Xb^*Yc^*$, then M deletes X and restarts;
5. if $\omega = a^*b^*Yc^*$, then M deletes Y and restarts;
6. finally, in all other situations M rejects.

Obviously, M iteratively repeats a series of four cycles and then accepts/rejects in a tail computation. From the above description, $L = L(M)$ and $L \in \mathscr{L}(\text{2-RLWW}(1)) \setminus \mathsf{CFL}$ follow easily. □

The previous theorem witnesses the insensitivity of input languages of RLWW(i)-automata with respect to the size of look-ahead window.

3.2 hRLWW(i)-Automata and h-Lexicalized Syntactic Disambiguation

In the following we will study basic and h-proper languages of hRLWW(i)-automata, and we will see that with respect to these languages, hRLWW(i)-automata (and their variants) are sensitive to the window size, number of SL-operations in a cycle, etc.

The reformulated basic result from [13] follows.

Theorem 6. *The following equalities hold:*

$$\begin{aligned} \mathsf{CFL} &= \mathscr{L}_{hP}(\text{mon-RLWW}(1)) \\ &= \mathscr{L}_{hP}(\text{det-mon-RLWW}(1)) \\ &= \mathscr{L}_{hP}(\text{det-mon-RLWWD}(1)) \\ &= \mathscr{L}_{hP}(\text{det-mon-RLWWC}(1)). \end{aligned}$$

The previous theorem witnesses that for any context-free language there is a deterministic monotone analyzer which accepts the language when each input word is completely lexically disambiguated.

3.3 Weak Cyclic Form for Basic Languages

Theorem 7. *Let $i \geq 1$. For each* RLWW*(i)-automaton M, there is a* wcf-RLWW*(i)-automaton M_{wcf} such that $L_{\mathrm{C}}(M) = L_{\mathrm{C}}(M_{\mathrm{wcf}})$, and $u \Rightarrow_M^{c^*} v$ implies $u \Rightarrow_{M_{\mathrm{wcf}}}^{c^*} v$, for all words u, v. Moreover, the added reductions are in contextual form. If M is deterministic, j-monotone or simultaneously deterministic and j-monotone, for some $j \geq 1$, then M_{wcf} is deterministic, j-monotone or simultaneously deterministic and j-monotone, respectively.*

Proof. **Transformation into wcf for basic languages.** Let M be an RLWW(i)-automaton. We describe how M can be converted into a wcf-RLWW(i)-automaton M_{wcf} with the basic language $L_{\mathrm{C}}(M_{\mathrm{wcf}}) = L_{\mathrm{C}}(M)$. Let us assume that the size of the window of M is k. It is easy to see that the language L_T accepted by M in tail computations is a regular sub-language of $L_{\mathrm{C}}(M)$. This means that there exists a deterministic finite automaton A_T such that $L(A_T) = L_T$. Assume that A_T has p states. For M_{wcf} we now take a window of size $k_{\mathrm{wcf}} = \max\{k, p+1\}$. M_{wcf} executes all cycles (reductions) of M just as M. However, the accepting tail computations of M are replaced by computations of M_{wcf} that work in the following way:

(1) Any word $w \in L_{\mathrm{C}}(M)$ satisfying $|w| \leq k_{\mathrm{wcf}}$ is immediately accepted.

(2) On any word w satisfying $|w| > k_{\mathrm{wcf}}$, M_{wcf} executes a cycle that works as follows: the window is moved to the right until the right sentinel \$ appears in the window. From the pumping lemma for regular languages we know that if $w \in L_T$, then there exists a factorization $w = xyz$ such that $|y| > 0$, $|y| + |z| \leq p$, and $xz \in L_T$. Accordingly, M_{wcf} deletes the factor y and restarts.

From the construction above we immediately see that $L_{\mathrm{C}}(M_{\mathrm{wcf}}) = L_{\mathrm{C}}(M)$. According to the definition of an RLWW(i)-automaton, the automaton M cannot rewrite during tail computations. Therefore, if M is deterministic then M_{wcf} is deterministic. Additionally, if M is j-monotone, then M_{wcf} is j-monotone, too, as the property of j-monotonicity is not disturbed by the delete operations at the very right end of the tape that are executed at the end of a computation. Moreover, all added reductions are in contextual form. □

This enables to strengthen Theorem 6 by requiring automata in weak cyclic form only.

Corollary 1. *For all $X \in \{\text{RLWW}(1), \text{RLWWD}(1), \text{RLWWC}(1)\}$, the following equalities hold:*

$$\mathsf{CFL} = \mathscr{L}_{hP}(\text{mon-wcf-}X) = \mathscr{L}_{hP}(\text{det-mon-wcf-}X).$$

Hence, we can require that analyzers for context-free languages are not only monotone, but they also should accept without restart short words only.

3.4 On Sensitivity of hLSA by hRLWW(i)-Automata

Here we will recall that for hRLWW(1)-automata in weak cyclic form, there are strict hierarchies of classes of finite and infinite basic and h-proper languages that are based on the window size and on the number of rewrites in one cycle (one reduction) the automata are allowed to execute in accepting computations. As the main new results we will show similar results for hRLWW(j)-automata.

First, however, we need to introduce some additional notions. For a type X of RLWW(j)-automata, we denote the subclass of X-automata which perform at most i reductions in any computation as fin(i)-X-automata, and by fin-X, we denote those X-automata that are of type fin(i)-X for some $i \geq 0$.

For any hRLWW(j)-automaton M, we use $L_{\mathrm{C}}(M,i)$ to denote the subset of $L_{\mathrm{C}}(M)$ that consists of all words that M accepts by computations that contain at most i reductions, and we take $L_{\mathrm{hP}}(M,i) = h(L_{\mathrm{C}}(M,i))$.

Proposition 8. *Let $i \geq 0$, $j \geq 1$ and let M be a* wcf-hRLWW(j)*-automaton. Then there exists a* fin(i)-wcf-hRLWW(j)*-automaton M_i such that $L_{\mathrm{C}}(M_i) = L_{\mathrm{C}}(M,i)$, M_i has the same window size as M, and if $u \Rightarrow_M^c v$ and $v \in L_{\mathrm{C}}(M,i-1)$, then $u \Rightarrow_{M_i}^c v$. In addition, if M is deterministic, then so is M_i.*

Proof. Obviously, for an arbitrary $i \geq 0$, $j,k \geq 1$ and k-fin(i)-wcf-hRLWW(j)-automaton M, the language $L_{\mathrm{C}}(M,i)$ is finite. Hence, it is accepted by a finite automaton A_i. Automaton M_i cannot simply accept in tail computations all words from $L_{\mathrm{C}}(M,i)$, as the words can be of length greater than k. Therefore, on input w, automaton M_i first simulates A_i. If A_i accepts, then let $v_1,\ldots,v_n$ be all words from $L_{\mathrm{C}}(M,i-1)$ such that $w \Rightarrow_M^c v_\ell$, for all $\ell, 1 \leq \ell \leq n$. Then M_i nondeterministically selects some ℓ_0 between 1 and n, and executes a cycle that rewrites w into v_{ℓ_0}. Automaton M_i must be able to execute a cycle $w \Rightarrow_{m_i}^c v_\ell$, for all $\ell, 1 \leq \ell \leq n$. This is possible, as both $L_{\mathrm{C}}(M,i)$ and $L_{\mathrm{C}}(M,i-1)$ are finite languages.

If M is deterministic, then $n = 1$ and M_i is deterministic, too. □

For a positive integer k, we will use the prefix de(k)- to denote those hRLWW-automata for which each reduction shortens the word on the tape by at most k symbols.

From the previous ITAT-contribution [14] we have the following hierarchies.

Theorem 9. *For all types $X \in \{$*hRLW(1), hRLWD(1), hRLWC(1), hRLWW(1), hRLWWD(1), hRLWWC(1)*$\}$, all prefixes $pr_1 \in \{\lambda, \mathsf{fin}(i), \mathsf{fin}\}$, where $i \geq 1$, all prefixes $pref_X \in \{$*wcf, mon-wcf, det-wcf, det-mon-wcf*$\}$, and all $k \geq 2$, we have the following proper inclusions:*

(a) $\mathscr{L}_{\mathrm{hP}}(k\text{-}pr_1\text{-}pref_X\text{-}X) \subset \mathscr{L}_{\mathrm{hP}}((k+1)\text{-}pr_1\text{-}pref_X\text{-}X)$,
(b) $\mathscr{L}_{\mathrm{hP}}(\mathsf{de}(k)\text{-}pr_1\text{-}pref_X\text{-}X) \subset \mathscr{L}_{\mathrm{hP}}(\mathsf{de}(k+1)\text{-}pr_1\text{-}pref_X\text{-}X)$,
(c) $\mathscr{L}_{\mathrm{A}}(k\text{-}pr_1\text{-}pref_X\text{-}X) \subset \mathscr{L}_{\mathrm{A}}((k+1)\text{-}pr_1\text{-}pref_X\text{-}X)$,
(d) $\mathscr{L}_{\mathrm{A}}(\mathsf{de}(k)\text{-}pr_1\text{-}pref_X\text{-}X) \subset \mathscr{L}_{\mathrm{A}}(\mathsf{de}(k+1)\text{-}pr_1\text{-}pref_X\text{-}X)$.

3.5 On Sensitivity of LSA by hRLWW(i)-Automata

As a simple consequence of Theorem 6 and the fact that the RLWW(1)-automaton M from the proof of Theorem 5(b) is actually deterministic we obtain the following corollary.

Corollary 2. *For all $i > 1$, $pref \in \{\lambda,$ det, wcf, det-wcf$\}$ and all $X \in \{$*hRLWW(i), hRLWWD(i), hRLWWC(i)*$\}$ the following holds:*

$$\mathsf{CFL} \subset \mathscr{L}_{hP}(pref\text{-}X).$$

The previous corollary and the following results strongly support the idea that hRLWW(i)-automata are strong and fine enough to cover and classify the complexity of the (surface and deep) syntactic features of natural languages such as subordination (dependency), valency, coordination etc.

Lemma 1. *For all $j,k \geq 1$ it holds the following:*

(1) $\mathscr{L}(k\text{-det-mon-fin}(1)\text{-wcf-RLWC}(j+1)) \setminus \mathscr{L}_{hP}(\text{k-wcf-hRLWW}(j)) \neq \emptyset$,

(2) $\mathscr{L}((k+1)\text{-det-mon-fin}(1)\text{-wcf-RLWC}(j)) \setminus \mathscr{L}_{hP}(\text{k-wcf-hRLWW}(j)) \neq \emptyset$.

Proof. To be able to show the lower bound we need a language containing word(s) that are longer than the window size.

For $r \geq 1$, $s \geq 0$ let $L^{(r,s)} = \{b, a^{(r-1)s}b^{s+1}\}$. In order to show $L^{(r,s)} \in \mathscr{L}(r\text{-det-mon-fin}(1)\text{-wcf-RLWC}(s))$ consider the mon-RLWW(s) automaton $M^{(r,s)}$ which on its left-to-right turn distinguishes three situations:

1. if input word is b, then $M^{(r,s)}$ accepts;
2. if input word is not from $L^{(r,s)}$, then $M^{(r,s)}$ rejects;
3. if input word is $a^{(r-1)s}b^{s+1}$, the automaton applies s CL operations each of which deletes $a^{r-1}b$; after that it restarts and accepts b in the next tail computation.

Realize that hRLWW(s) automaton with window size r can delete at most rs symbols in any cycle with at most s rewrite (W- or SL-) operations. Since $|a^{(r-1)s}b^{s+1}| - |b| = rs$, neither wcf-hRLWW($s$) automaton with window size $r-1$ nor wcf-hRLWW($s-1$) automaton with window size r is able to recognize $L^{(r,s)}$ as its basic or h-proper language.

Now, the languages $L^{(k,j+1)}$ and $L^{(k+1,j)}$ can be used to prove (1) and (2), respectively. □

Next we show a similar hierarchy with respect to the degree on monotonicity which is related to the number of rewritings in a cycle. The degree of monotonicity is an important constraint which serves for better approximation of syntax of natural languages, and their individual features.

Lemma 2. *For all $j,k \geq 1$ it holds the following:*

(1) $\mathscr{L}(k\text{-det-mon}(j+1)\text{-wcf-RLWWC}(j+1)) \setminus \mathscr{L}_{\mathrm{hP}}(k\text{-wcf-RLWW}(j)) \neq \emptyset$;

(2) $\mathscr{L}(k\text{-det-mon}(j+1)\text{-wcf-RLWWC}(j+1)) \setminus \mathscr{L}_{\mathrm{hP}}(k\text{-mon}(j)\text{-wcf-RLWW}(j+1)) \neq \emptyset$.

Proof. We provide a parametrized sequence of finite languages $\left\{L_{\mathrm{m}}^{(k,j)}\right\}_{j,k=1}^{\infty}$, which satisfy

(a) $L_{\mathrm{m}}^{(k,j+1)} \in \mathscr{L}(k\text{-det-mon}(j+1)\text{-wcf-RLWC}(j+1))$,
(b) $L_{\mathrm{m}}^{(k,j+1)} \notin \mathscr{L}_{\mathrm{hP}}(k\text{-wcf-hRLWW}(j))$, and
(c) $L_{\mathrm{m}}^{(k,j+1)} \notin \mathscr{L}_{\mathrm{hP}}(k\text{-mon}(j)\text{-wcf-RLWW}(j+1))$.

Let $L_{\mathrm{m}}^{(k,j)} = \{(c^k d^k e^{k(j+1)})^j, (d^k e^{k(j+1)})^j\} \cup \{e^{krj} \mid 0 \leq r \leq j+1\}$. We can construct a k-det-mon(j)-RLWC(j)-automaton $M_{\mathrm{m}}^{(k,j)}$ accepting $L_{\mathrm{m}}^{(k,j)}$ as its input (and basic) language. The automaton scans the word on its tape and distinquishes the following cases:

1. if the word is of the form $(c^k d^k e^{k(j+1)})^j$, then $M_{\mathrm{m}}^{(k,j)}$ deletes j blocks of c^k and restarts;
2. if the word is of the form $(d^k e^{k(j+1)})^j$, then $M_{\mathrm{m}}^{(k,j)}$ deletes j blocks of d^k and restarts;
3. if the word is of the form e^{krj}, for some $r, 1 \leq r \leq j+1$, then $M_{\mathrm{m}}^{(k,j)}$ deletes j blocks of e^k and restarts;
4. if the word is empty, then $M_{\mathrm{m}}^{(k,j)}$ accepts, and
5. otherwise, $M_{\mathrm{m}}^{(k,j)}$ rejects.

It is not hard to partition the right distances of the rewriting configurations from an accepting computation of $M_{\mathrm{m}}^{(k,j)}$ into at most j monotone sequences. The highest degree j of monotonicity is necessary for accepting the input word $(c^k d^k e^{k(j+1)})^j$, which is accepted by $M_{\mathrm{m}}^{(k,j)}$ after performing $j+3$ reductions.

For reasons similar to that ones used in the proof of Lemma 1, the language $L_{\mathrm{m}}^{(k,j)}$ cannot be accepted neither as an h-proper language of a k-wcf-RLWW$(j'-1)$-automaton for any $j' < j$ nor as an h-proper language of a k-mon(j')-wcf-RLWW(j)-automaton, for any $j' < j$. □

The sample languages from the proofs of Lemma 1 and Lemma 2 have disjunctive alphabets and, therefore, they can be combined in order to obtain hierarchies with combined constraints. E.g., the language $L^{(k,j)} \cup L_{\mathrm{m}}^{(k',j')}$ is a language which can be accepted as an h-proper language of a $\hat{k}$-det-mon(j')-wcf-RLWC$(\hat{j})$-automaton, where $\hat{k} = max(k,k')$ and $\hat{j} = max(j,j')$, but not as an h-proper language of neither a k-wcf-RLWW$(j-1)$-automaton, nor a k'-mon$(j'-1)$-wcf-RLWWC(j').

In a similar way, we obtain the following consequences. We present here only such consequences which can be interpreted as linguistically relevant with respect to the complete lexical disambiguation.

Corollary 3. *For all $j,k \geq 1$, all prefixes $pref_X, pref_Y \in \{$det-wcf, det-fin(i)-wcf$\}$ and all $X,Y \in \{$hRLWW, hRLWWD, hRLWWC$\}$, the following holds:*

(a) $\mathscr{L}_A(k\text{-}pref_X\text{-}X(j)) \subset \mathscr{L}_A(k\text{-}pref_X\text{-}X(j+1))$,
(b) $\mathscr{L}_A(k\text{-}pref_X\text{-}X(j+1)) \setminus \mathscr{L}_A(k\text{-}pref_Y\text{-}Y(j)) \neq \emptyset$,
(c) $\mathscr{L}_A(k\text{-}pref_X\text{-}X(j)) \subset \mathscr{L}_A((k+1)\text{-}pref_X\text{-}X(j))$,
(d) $\mathscr{L}_A((k+1)\text{-}pref_X\text{-}X(j)) \setminus \mathscr{L}_A(k\text{-}pref_Y\text{-}Y(j)) \neq \emptyset$.

If additionally $i \geq j+3$, the following holds:

(aa) $\mathscr{L}_A(k\text{-mon}(j)\text{-}pref_X\text{-}X(j)) \subset \mathscr{L}_A(k\text{-mon}(j+1)\text{-}pref_X\text{-}X(j+1))$,
(bb) $\mathscr{L}_A(k\text{-mon}(j+1)\text{-}pref_X\text{-}X(j+1)) \setminus \mathscr{L}_A(k\text{-}pref_Y\text{-}Y(j)) \neq \emptyset$,
(cc) $\mathscr{L}_A(k\text{-mon}(j)\text{-}pref_X\text{-}X(j)) \subset \mathscr{L}_A((k+1)\text{-mon}(j)\text{-}pref_X\text{-}X(j))$,
(dd) $\mathscr{L}_A((k+1)\text{-mon}(j)\text{-}pref_X\text{-}X(j)) \setminus \mathscr{L}_A(k\text{-}pref_Y\text{-}Y(j)) \neq \emptyset$.

We can see that the h-lexicalized syntactic analyses of det-wcf-hRLWW(j)-automata are sensitive to the maximal number of rewrite (SL- and W-) operations in a cycle and to the size of the window. The syntax of these languages is given directly by individual reductions, i.e., by individual instructions of the hRLWW(j)-automata. Namely the reductions of RLWWC(j)-automata describe (define) the (discontinuous) syntactic constituents of the analyzed words (sentences). Note that the monotonicity of degree one means a synonymy for context-freeness of the accepted languages by restarting automata. The monotonicity of higher degree means a degree of non-context-freeness of accepted languages. In the previous corollary we have transferred this concept from infinite to finite languages. That can be very useful for classification of individual syntactic features.

4 Conclusion

We have seen that monotone RLWW(i)-automata are not sensitive to the number of deletions and to the size of their window with respect to their input languages and that these languages do in general not yield reduction correctness preserving computations of RLWW(i)-automata. On the other hand, hRLWW(i)-automata satisfy the reduction correctness preserving property with respect to their basic and h-proper languages, and consequently also with respect to their lexicalized syntactic analysis. The reduction

correctness preserving property enforces the sensitivity to the number of rewritings in a reduction and to the size of the window.

We believe that the class of h-proper languages of det-mon(2)-wcf-RLWWC(2)-automata is strong enough to model lexicalized (surface) syntax of natural languages, that is, to model their reduction correctness preserving lexicalized syntactic analysis. Namely, we strongly believe that the class of h-proper languages of det-mon(2)-wcf-RLWWC(2)-automata is a superclass of the class of mildly context-sensitive languages [5, 6]. In the future we will try to characterize the class of mildly context-sensitive languages by h-proper languages of RLWW-automata with some constraints.

Our long term goal is to propose and support a formal (and possibly also software) environment for a further study and development of Functional Generative Description (FGD) of Czech (see [9]). We strongly believe that the lexical disambiguation of FGD can be fully described by (a certain refinement of) det-mon(4)-wcf-RLWWD(4)-automata.

We stress that our current efforts cover an important gap in theoretical tools supporting computational and corpus linguistics. Chomsky grammars and the corresponding types of automata do not support lexicalized syntactic analysis, as these grammars work with categories bound to individual constituents with respect to constituent syntactic analysis. They do not support syntactic analysis with any kind of correctness preserving property, but they do support several types of insensitivity to the form of individual grammar rules (see several normal forms for context-free grammars, like Chomsky and Greibach normal form [2]), and, finally, they do not support the classification of finite phenomena of (natural) languages.

On the other hand, in corpus linguistics, only finite language phenomena can be observed. Now the basic and h-proper languages of hRLWW(i)-automata in weak cyclic form allow common classifications of finite phenomena as well as classifications of their infinite generalizations to the corresponding parts of the Chomsky hierarchy. All these classifications are based on the reduction correctness preserving property and the weak cyclic form. Let us recall for restarting and list automata the monotonicity means a synonymy for context-freeness. Here we are able to distinguish between finite monotone and finite non-monotone languages (syntactic phenomena), too.

Finally, note that many practical problems in computational and corpus linguistic became decidable if we consider only parametrized finite languages.

References

[1] Yehoshua Bar-Hillel: A quasi-arithmetical notation for syntactic description. *Language* 29: 47–58 (1953)

[2] John E. Hopcroft, Jeffrey D. Ullman: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Reading, M.A. (1979)

[3] Petr Jančar, František Mráz, Martin Plátek, Jörg Vogel: Restarting automata. In: Horst Reichel (ed.): FCT'95, Proc., pages 283–292, LNCS 965, Springer, Berlin (1995)

[4] Petr Jančar, František Mráz, Martin Plátek, Jörg Vogel: On monotonic automata with a restart operation. *J. Autom. Lang. Comb.* 4: 287–311 (1999)

[5] Aravind K. Joshi, K. Vijay-Shanker, David Weir: The convergence of mildly context-sensitive grammatical formalisms. In: Peter Sells, Stuart Shieber, Tom Wasow (eds.), *Foundational Issues in Natural Language Processing*, pages 31–82, MIT Press, Cambridge MA (1991)

[6] Aravind K. Joshi, Yves Schabes: Tree-adjoining grammars. In: Grzegorz Rozenberg, Arto Salomaa (eds.), *Handbook of Formal Languages*, vol. 3, pages 69–123, Springer, Berlin, New York (1997)

[7] Tomasz Jurdziński, František Mráz, Friedrich Otto, Martin Plátek: Degrees of non-monotonicity for restarting automata. *Theor. Comp. Sci.* 369: 1–34 (2006)

[8] Markéta Lopatková, Martin Plátek, Vladislav Kuboň: Modeling syntax of free word-order languages: Dependency analysis by reduction. In: Václav Matoušek, Pavel Mautner, Tomáš Pavelka (eds.), TSD 2005, Proc., pages 140–147, LNCS 3658, Springer, Berlin (2005)

[9] Markéta Lopatková, Martin Plátek, Petr Sgall: Towards a formal model for functional generative description: Analysis by reduction and restarting automata. *Prague Bull. Math. Linguistics* 87: 7–26 (2007)

[10] František Mráz, Martin Plátek, Friedrich Otto: A Measure for The Degree of Nondeterminism of Context-free Languages. In: Jan Holub and Jan Žďárek (eds.), *Proceedings of CIAA 2007*, pages 192–20S, LNCS 4783, Springer, Berlin (2007)

[11] Niemann, G. Otto, F.: Restarting automata, Church-Rosser languages, and representations of re languages. In: Developments In Language Theory: Foundations, Applications, and Perspectives, World Scientific, 2000, 103–114

[12] Martin Plátek: Two-way restarting automata and j-monotonicity. In: Leszek Pacholski, Peter Ružička (eds.): SOFSEM'01, Proc., pages 316–325, LNCS 2234, Springer, Berlin (2001)

[13] Martin Plátek, Friedrich Otto: On h-lexicalized restarting automata. In: Erzsébet Csuhaj-Varjú, Pál Dömösi, György Vaszil (eds.), *AFL 2017, Proc.*, Open Publishing Association, EPTCS 252: 219–233 (2017), DOI:10.4204/EPTCS.252.21

[14] Martin Plátek, Friedrich Otto, František Mráz: On h-lexicalized automata and h-syntactic analysis. In: *ITAT 2017, Proc.*, CEUR Workshop Proceedings Vol. 1885, pp. 40–47 (2017)

[15] Martin Plátek, Friedrich Otto, František Mráz: On h-Lexicalized Restarting List Automata. *Technical Report*, www.theory.informatik.uni-kassel.de/projekte/RL2016v6.4.pdf, Kassel (2017)

[16] Natalie Schluter: Restarting automata with auxiliary symbols restricted by lookahead size. *Intern. J. Comput. Math.* 92: 908–938 (2015)

S. Krajči (ed.): ITAT 2018 Proceedings, pp. 18–27
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Michal Štrba and Richard Ostertág

Approaching side-effects in pure functional programming by interpreting data structures as recipes

Michal Štrba, Richard Ostertág

Faculty of Mathematics, Physics and Informatics, Comenius University, Mlynská dolina, Bratislava, Slovakia
faiface@ksp.sk, ostertag@dcs.fmph.uniba.sk

Abstract: We present a new approach to side-effects in pure functional programming. The approach is not novel in every aspect but rather in combining of multiple already known concepts into a coherent method.

The key concept in this approach is to view some data structures as describing a recipe of what should be done. A program expressed in form of this data structure is then passed to another program called 'interpreter'. An interpreter takes the recipe, examines it, and actually executes all the desired side-effects.

Keeping simplicity, clarity, and avoiding introduction of any unnecessary theory, the recipe data structures tend to roughly conform to the CPS (continuation passing style) model and tend to contain function values inside of them. This means, that expressing them in code involves a lot of anonymous functions and large "continuations" as arguments to other functions. Attempting to write such code in any of the traditional functional languages, most notably Haskell, results in a very messy, unpleasant, and hard to read code. We also think that it's the main reason why techniques described in this paper weren't developed earlier in these languages.

Therefore, we created a new language that makes such code beautiful and pleasant to read. The subtle features of the language that make this new approach to side-effects viable have far reaching implications, mostly by making it possible to write purely functional code that reads top to bottom.

In this paper, we describe the Funky programming language, how it facilitates writing vertical code, how we can write interpreters for recipe data structures, and how we can use the language to transform and combine the recipes in purely functional code and thereby attain great expressivity in writing side-effecting programs.

1 Short introduction to Funky

Funky is a simple language with a small set of orthogonal features that combine very well. The language was designed and implemented by us and was the topic of our bachelor's thesis. This paper is a shortened version of that thesis. The parts left out in this paper are mainly the thorough description of the language. This shouldn't cause any trouble to people generally familiar with functional programming, for whom this paper is intended anyway.

Now we'll briefly describe the important aspects of the language so that the following sections are easy to understand. The description is very dense, full comprehension is not required.

1.1 Names and tokens

Tokens in Funky are generally separated by whitespace, except for these special characters which are always parsed as separate tokens, whether separated by whitespace or not:

```
( ) [ ] { } , ; \ #
```

Aside from these, all tokens are separated by whitespace. Consequentially, identifiers may contain all kinds of symbols. For example, these are all valid identifiers: `fold>`, `fold<`, `empty?`, `skip-whitespace`. Dashes (`-`) are used to separate words in function names instead of underscores or camel-case. However, type names start with upper-case letters and use camel-case.

Tokens starting with a digit or a +/- sign followed by a digit are numbers (valid or invalid). Any series of characters enclosed in single quotes is a character literal (valid or invalid) and similarly, double quotes enclose a string literal (again, valid or invalid). Escaping in characters/strings works as expected.

1.2 Functions

A function definition is signified by the `func` keyword, which is followed by a function name, a colon, the type of the function, equals sign and finally the function body – an expression. Function definitions cannot be nested and expressions are only allowed as function bodies, no expressions outside functions are meaningful. For example:

```
func sqrt : Float -> Float = \x x ^ 0.5

func max : Int -> Int -> Int =
    \x \y if (x >= y) x y

func flip : (a -> b -> c) -> b -> a -> c =
    \f \x \y
    f y x
```

As we can see, function argument names are not specified before the equals sign, instead, all arguments are introduced by abstractions. Funky is very consistent with this – literally all variables in expressions are introduced by abstractions, there isn't a single case otherwise.

Abstractions are very concise – they consist only of a backslash followed by the name of the bound variable followed by the body of the abstraction (function). Multiple arguments are introduced simply by nesting abstractions. The body of an abstraction always spans until the end of the scope (for example, inside parentheses it spans until the closing parenthesis), which prevents unnecessary parentheses in many cases.

Function names consisting solely of special characters (no letters or numbers) are infix functions. All infix functions have the same precedence, which is lower than the precedence of regular prefix function application and are all right-associative. This is to free the programmer from the burden of manually specifying the precedence and associativity of infix functions. Right-associativity was chosen because it's more often useful than left-associativity.

The type system is very similar to the one in Haskell. The main difference is that Funky has no type-classes and disallows higher-kinded types (type variables with arguments), which shall not be confused with higher-order types that are supported in Funky.

Quite unusually and very importantly, Funky supports function overloading – defining multiple functions with the same name, but different types. For example:

```
# map applies a function to all elements of
# a list
func map : (a -> b) -> List a -> List b =
    \f
    fold< ((::) . f) []  # :: is cons

# map applies a function to the potential
# content of a maybe
func map : (a -> b) -> Maybe a -> Maybe b =
    \f \maybe
    switch maybe
    case nothing nothing
    case just \x just (f x)

# useless converts a list of maybe floats to
# a list of maybe ints
# ^ that's a useless comment
func useless : List (Maybe Float) ->
List (Maybe Int) =
    map (map int)
```

This comes very handy in many situations – programmer doesn't have to think about names too much and similar behavior on different types may be assigned the same function name. Also, as we'll see, records are thereby allowed to share field names, which is something that causes a lot of trouble in Haskell.

Function overloading is not arbitrary – overloaded functions with colliding types are disallowed. Colliding types are such types that can be specialized (their type variables can be substituted) into the same type. This is because it would be impossible to determine, or even easily express, which of the colliding functions should be used in many situations.

In the case of need or a desire for clarification, any expression can be type-annotated with a colon:

```
((x : Int) + (y : Int) : Int)
```

1.3 Records

Records are one of the three means of creating own types in Funky (the other two are unions and aliases). Records are similar to structs from C or records from Pascal. They are compound types, a single value containing multiple fields.

A record definition is signified by the `record` keyword, which is followed by the record name, a list of type variables required by the record (if any), and an equals sign followed by a comma-separated list of fields. All fields must be type annotated.

```
record Pair a b = first : a, second : b

record Person =
    name : String,
    age  : Int,  # trailing comma is allowed

record Vec4D =
    x : Float,
    y : Float,
    z : Float,
    w : Float,
```

Funky generates a few functions per record: the constructor and a getter and an updater for each field. For example, in case of the `Person` record, these functions get generated:

```
func Person : String -> Int -> Person

func name : Person -> String
func name : (String -> String) -> Person ->
    Person

func age : Person -> Int
func age : (Int -> Int) -> Person -> Person
```

The constructor takes the values of the record fields in order and returns an instance of the record. A getter simply takes a record value and returns the given field. An updater is more peculiar. It takes a function mapping the record field to a new value and a record value. Then it returns a copy of the original record where the given field is replaced by the result of applying the function to its original value. This approach to updaters was chosen for two reasons: first is that this is usually what we want to do: to update a field according to its previous value; the second reason is that updaters compose very well in this form (they were inspired by Lenses from Haskell).

For example, let's work with these two records:

```
record Point = x : Int, y : Int
```

```
record Segment =
    start : Point,
    end   : Point,
```

Say we have a variable `seg` which is a segment. We can compose getters to access the X coordinate of the starting point:

```
(x . start) seg
```

But we can also compose updaters to change the value of that coordinate and get an updated segment:

```
(start . x) (+ 4) seg
```

To replace the value of a field with a value independent of the previous value of the field, we use the `const` function (`const x` takes one argument and always returns `x`):

```
(start . x) (const 0) seg
```

Let's define one more record:

```
record Plan = segments : List Segment
```

The `map` function can also be used as an updater on a list, and so we can write this to update all Y coordinates of all the end points of the segments in a plan:

```
(segments . map . end . y) (* 2) plan
```

The whole main motivation for the creation of the huge and abstract Lens library in Haskell is avoided in Funky simply by providing clever updater functions.

1.4 Unions

Unions are just like data types in Haskell. Their definition is signified by the `union` keyword followed by the name of the union and a list of type variables, then an equals sign followed by a `|` separated list of alternative forms. For example (`::` is cons, the list prepend function):

```
union Bool = true | false

union Maybe a = nothing | just a

union List a = empty | a :: List a
```

Funky generates a single constructor function for each alternative, which takes all the arguments in the order and returns an instance of the union. For example, these functions get generated for the `List` union:

```
func empty : List a
func (::)  : a -> List a -> List a
```

To get values out of a union value, Funky introduces a special `switch/case` structure that looks like this:

```
func length : List a -> Int =
    \list
    switch list
    case empty
        0
    case (::) \x \xs
        1 + length xs
```

Each case is followed by the name of the alternative, which is followed by a function that accepts the arguments of the alternative and returns the final result of the `switch/case` structure. The arguments in the case body don't have to be mentioned explicitly – the above function could've been written like this:

```
func length : List a -> Int =
    \list
    switch list
    case empty 0
    case (::)  const ((1 +) . length)
```

1.5 Aliases

The simplest way of making type names is by aliasing. Alias simply defines another name for an existing type, albeit a possibly more complex one. For example, the `String` type from the standard library is defined like this:

```
alias String = List Char
```

The alias is perfectly identical to the type on right side of the equals side. They may be used interchangeably. Aliases may also have type variables.

2 Vertical code

No feature is useful unless the benefits of its employment outweigh the costs – the feature must be viable. The new approach to side-effects described in this paper is viable in Funky, but not in Haskell, or any other traditional purely functional language. What makes the difference? Surprisingly, only two subtle syntactic differences, nothing major at the first sight. Yet, these two syntactic differences make code that is otherwise ugly and messy in Haskell beautiful and readable in Funky. Of course, some code that is beautiful and readable in Haskell is in its original form not possible in Funky at all. But, we're not talking about just any code here: we're talking about the kind of code that makes the new approach to side-effects viable and that is – vertical code.

Writing code that reads top to bottom and composes vertically has always been the domain of imperative programming. "Do this, then do that, then repeat this until some other thing happens" is a very natural way of expressing programs. Programmer reads the code top to bottom, remembering the invariants as they occur, until they form a complete picture of the program in their mind.

Functional programs tend to compose differently. Instead of a series of statements, functional programs are mere expressions constructed mostly from function application and abstraction. An expression has no natural "order of execution", it's usually a function application whose arguments are yet other expressions. The arguments can be understood in any order. However, if the arguments are large expressions containing more function applications with still more large expressions as their arguments, the whole thing becomes very inconvenient to read and understand. The reason is that humans operate with only a finite (and fairly small) amount of working memory. Understanding an expression that is syntactically a wide and deep tree requires a lot of working cognitive memory, so it's hard.

Of course, designers of functional languages are well aware of this. Many syntactic features present in those languages are specially designed to tackle this issue. These features include: pattern matching, guards, let bindings, where bindings, or Haskell's do notation. However, none of these features solves the problem fully, because they don't nest very well.

The motivation for designing Funky arose from the harmony experienced while playing with the pure λ-calculus and from the frustration with existing functional languages. Experimenting with the pure λ-calculus gave us the opportunity to step back – see the bigger picture. We saw that instead of adding new syntactic features like the ones described above, all that was needed was to improve the core – make writing ordinary expressions more concise and make it possible to naturally split them into multiple lines. It turned out that this was enough. Funky has no pattern matching, guards, let binding syntax, nor anything analogous to the Haskell's do notation.

The two subtle syntactic features that make it possible are: concise trailing lambdas and the semicolon. We'll demonstrate them on concrete examples: the `if` and the `let` function in Funky.

2.1 `if` and `let`

In Funky, `if` is a simple function from the standard library (body omitted):

```
func if : Bool -> a -> a -> a
```

It takes a condition and two arguments of the same type: then and else, and returns back the correct one. It can be used as a simple conditional expression, just like the if structure in Haskell, or the ternary operator in C:

```
func min : Int -> Int -> Int =
    \x \y
    if (x < y) x y
```

But if the expressions are more complex, this becomes hard to read. Let's take the factorial function as an example (we'll call it `n!` because Funky allows this):

```
func n! : Int -> Int =
    \n
    if (n <= 0) 1 (n * n! (n - 1))
```

This particular code isn't too bad, but it's easy to conceive situations where using the `if` as a simple, one-line conditional expression would be outright unacceptable.

Funky introduces a special syntactic concept: the semicolon. All it does is it puts everything that's after it (in the scope) inside parentheses. It works just like the `$` function in Haskell. So we can rewrite this:

```
if (n <= 0) 1 (n * n! (n - 1))
```

into this:

```
if (n <= 1) 1; n * n! (n - 1)
```

and then split it into multiple lines:

```
if (n <= 1)
    1;
n * n! (n - 1)
```

A more involved example is the infamous FizzBuzz problem. Here's a function that returns the correct output for each number:

```
func fizzbuzz : Int -> String =
    \number
    if ((number % 15) == 0)
        "fizzbuzz";
    if ((number % 3) == 0)
        "fizz";
    if ((number % 5) == 0)
        "buzz";
    string number
```

Here, `if` is used to express a series of cases terminated by a catch-all case. The built-in if structure in Haskell is not suitable for such purposes – one would have to use guards instead. But as we've already said, guards don't nest so well.

In contrary, Funky's `if` function nests perfectly. Both "then" and "else" expressions may be arbitrarily large, nested, vertical expressions.

```
if condition (
    then
    ...
);
else
...
```

Similarly to `if`, let bindings (variable assignments) are not a built-in language construct in Funky. Instead, `let` is a function from the standard library. Here's its full definition (including the body):

```
func let : a -> (a -> b) -> b = \x \f f x
```

The function `let` takes a value and then takes a function to which it passes the value as an argument. Thanks to the Funky's concise function construction (lambda/backslash), this function is a perfectly viable replacement for a let binding construct built directly into the language.

```
let "me stay by your"
(\side reverse side ++ " " ++ side)
```

Here we call the `let` function with two arguments: the first one is the string `"me stay by your"` and the second one is a function. Calling `let` passes the string as the argument to the function and the whole expression evaluates to `"ruoy yb yats em me stay by your"`.

Thanks to the Funky's trailing lambda syntax (function body spans until the end of scope), we can remove the parentheses around the function:

```
let "me stay by your"
\side reverse side ++ " " ++ side
```

Furthermore, we can split the expression into two lines, yielding a very readable piece of code:

```
let "me stay by your" \side
reverse side ++ " " ++ side
```

Now, instead of thinking in terms of function applications and abstractions, we understand the code as an assignment to a variable (immutable, of course) followed by the resulting expression.

To assign multiple variables, we can just stack `let` assignments:

```
let (filter prime? (count 2))   \primes
let (take-while (< 100) primes) \small-primes
let (length small-primes)       \count
string count ++ " small primes (< 100)"
```

This expression evaluates to: `"25 small primes (< 100)"`.

2.2 Vertical functions

Some function are suitable for composing code vertically, either with the help of the semicolon or trailing lambdas. We've seen both in the previous section on the `if` and the `let` function. Other functions aren't suitable for that. Now we'll describe a small theory about these functions that allow vertical composition. We call them *vertical functions*.

There are two main kinds of vertical functions: those that utilize the semicolon and those that involve trailing lambdas. Both are characterized by their signature (type).

The first kind we call *semicolon kind vertical functions*. Their signatures have this general form (where `V` is an arbitrary type and `...` is an arbitrary sequence of `->` applications):

```
... -> V -> V
```

The last argument to a semicolon kind vertical function can be usually understood as a *continuation*, analogous to CPS (continuation passing style).

The `if` function is an example of a semicolon kind vertical function.

In some cases, the above form is violated and the return type doesn't match the last argument, while the function is still used vertically. However, the signature usually follows the mentioned form.

The second kind we call *trailing lambda kind vertical functions*. Their signatures have this general form:

```
... -> (... -> V) -> V
```

The vertical function passes some arguments to the continuation, which then takes on the same role as with the semicolon kind vertical functions. Again, some vertical functions may violate this form, but those cases are rare.

The `let` function is an example of a trailing lambda kind vertical function.

2.3 Viability

The main reason why vertical functions along with their usage haven't seen the light of the day in traditional functional languages, such as Haskell is that they're hardly viable there. The reasons are very subtle. After all, Haskell supports anonymous functions and has the `$` function analogous to the Funky's semicolon.

To demonstrate this, let's take a function for generating all permutations of a list (the code contains some functions we haven't discussed, and won't discuss, in this paper, but the understanding of what the function does is irrelevant now):

```
func permutations : List a -> List (List a) =
    \list
    if (empty? list)
        (yield []; empty);
    pick (permutations (rest list)) \tail
    pick (insert (first list) tail) \perm
    yield perm;
    empty
```

In Haskell, the above code could be rewritten roughly like this:

```
permutations list =
 if null list
  then yield [] $ empty
  else
   pick (permutations (rest list)) (\tail ->
   pick (insert (first list) tail) (\perm ->
   yield perm $
   empty))
```

In case `if` was a function in Haskell, instead of a built-in construct, we can improve the code a bit:

```
permutations list =
 if (null list)
   (yield [] $ empty) $
 pick (permutations (rest list)) (\tail ->
 pick (insert (first list) tail) (\perm ->
 yield perm $
 empty))
```

We could improve the code a little bit more by removing the parentheses around anonymous functions and inserting a few more dollars:

```
permutations list =
 if (null list)
   (yield [] $ empty) $
 pick (permutations (rest list)) $ \tail ->
 pick (insert (first list) tail) $ \perm ->
 yield perm $
 empty
```

The result is still fairly bad, though. The dollar signs all over the place stick out too much and the arrow at the end of the argument list of an anonymous function is similarly detrimental to the overall aesthetics. It's no wonder that vertical functions weren't in fact invented in Haskell. Of course, one would use pattern matching or the do notation to write a similar function in Haskell. However, neither of those features nests very well. The approach taken by Funky is more general.

3 Side-effects and interpreters

Funky's approach to side-effects is unique among functional languages, but after exploring it, this fact comes rather surprising. The approach is so obvious that it's quite curious no other language (to our knowledge) has adopted it before.

The idea is this: a program in Funky is just a value, a data structure. Then there is a special program called *interpreter*. This program interacts with the Funky's runtime (evaluator) to examine this data structure, which serves the role of a recipe. A recipe then tells the interpreter what to do. Contrary to ordinary recipes, these recipes are generated, combined, recursive, and so on, all the goods of functional programming.

The Funky programming language itself has no intrinsic concept of side-effects. In fact, the interpreters themselves add no real concept of side-effects either. Everything still remains just a value. The recipe is a value that can be transformed and manipulated just like any other value. This is where a lot of expressive power comes in as we'll see.

There isn't just one interpreter. In fact, any Funky programmer can make their own interpreters for their own special purposes. Each interpreter works with a different data structure describing the desired side-effects. One interpreter is intended for command-line applications. Another one is for web servers. And yet another is for 2D games. Each interprets a data structure specialized for the given task. In this paper, we'll examine the interpreter for command-line applications.

3.1 Interpreters

Funky's runtime is currently written in Go. As interpreters need to interact with the runtime, they too must be written in Go. This may be expanded to more languages in the future, for example, it would be useful to write Funky interpreters in Java or C++.

To write an interpreter we need to import the `"github.com/faiface/funky"` package and call the `funky.Run` function. This function does all the job regarding command-line flags, reading, parsing, and compiling source files, and returns a runtime value representing the data structure, the recipe, back to the programmer.

```
package main

import "github.com/faiface/funky"

func main() {
    program := funky.Run("main")
}
```

The `funky.Run` function takes one argument: the name of the function containing the recipe value. The return type of the `funky.Run` function is `*runtime.Value`. The `runtime` package is located at `"github.com/faiface/funky/runtime"`. The `*runtime.Value` type provides several methods we can use to interact with the value:

```
func (*runtime.Value) Char() rune
func (*runtime.Value) Int() *big.Int
func (*runtime.Value) Float() float64
func (*runtime.Value) Field(i int)
    *runtime.Value
func (*runtime.Value) Alternative() int
func (*runtime.Value) Apply(
    arg *runtime.Value) *runtime.Value

// these three are implemented using the
// above six
func (*runtime.Value) Bool() bool
func (*runtime.Value) List() []*runtime.Value
func (*runtime.Value) String() string
```

The `Char`, `Int`, and `Float` methods are used to retrieve values of Funky's built-in types. The `Field` function returns the `i`-th field of a record, or the `i`-th argument to a union constructor. The `Alternative` method returns the index of the union constructor of the value. And lastly, the `Apply` function takes another runtime value and applies it to the function in the receiving runtime value and returns the result of this application.

All the above functions crash if they're used on values of wrong types. For example, calling `Alternative` on a record value crashes, and calling `Char` on a float value likewise.

The last three functions are just for convenience because booleans, lists, and strings are very widely used types.

The interpreter sometimes needs to fabricate new runtime values not originating in the Funky program. For example, when a command-line interpreter loads a character from the input, it needs to make a character value and pass it to the program. These functions are provided in the "github.com/faiface/funky/runtime" package for this purpose:

```
func MkChar(c rune) *runtime.Value
func MkInt(i *big.Int) *runtime.Value
func MkFloat(f float64) *runtime.Value
func MkRecord(fields ...*runtime.Value)
    *runtime.Value
func MkUnion(alt int, fields ...*runtime.Value)
    *runtime.Value

// these three are, again, implemented using
// the above five
func MkBool(b bool) *runtime.Value
func MkList(elems ...*runtime.Value)
    *runtime.Value
func MkString(s string) *runtime.Value
```

Their meaning is clear, so we'll avoid explaining that.

3.2 Interactive command-line programs

The data structure serving the role of a recipe we chose for simple interactive command-line programs is strikingly simple:

```
union IO =
    done               |
    putc Char IO       |
    getc (Char -> IO) |
```

It is a kind of a linked list, with three types of nodes. One signals the end of the program: `done`. The next one – `putc` – says that a character should be printed and the program should continue in some way. The first argument to `putc` is the character to be printed. The other argument is the rest of the program – a continuation. The last node – `getc` – is requesting a character from the input. It has one argument: a function. The interpreter should read the character and pass it as an argument to this function. The function then returns the rest of the program.

Funky always generates constructor functions for a union and these are the ones generated for `IO` (bodies omitted, internal to the compiler/runtime):

```
func done : IO
func putc : Char -> IO -> IO
func getc : (Char -> IO) -> IO
```

The imporant thing to notice is that `putc` is a semicolon kind vertical function and `getc` is a trailing lambda kind vertical function. This makes it easy to compose interactive command-line programs in a natural, imperative-like style.

For example, here's a "cat" program, a program that simply copies the input to the output:

```
func main : IO =
    getc \c
    putc c;
    main
```

This program has no `done` node, it's an infinite data structure. Before we run it, though, we need an interpreter. Here it is (badly formatted, because the lack of horizontal space):

```
package main

import (
    "bufio"
    "io"
    "os"
    "github.com/faiface/funky"
    "github.com/faiface/funky/runtime"
)

func main() {
    program := funky.Run("main")
    in := bufio.NewReader(os.Stdin)
    out := bufio.NewWriter(os.Stdout)
    defer out.Flush()
loop:
    for {
        switch program.Alternative() {
        case 0: // done
            break loop
        case 1: // putc
            out.WriteRune(program.Field(0).Char())
            program = program.Field(1)
        case 2: // getc
            out.Flush()
            r, _, err := in.ReadRune()
            if err == io.EOF {
                break loop
            }
            program = program.Field(0).
                        Apply(runtime.MkChar(r))
        }
    }
}
```

The interpreter enters a loop where it checks the program node type and acts accordingly, always proceeding to the continuation until reaching the `done` node.

Now we can run the "cat" program (user input is emphasized, `funkycmd` is the name of the interpreter):

```
$ funkycmd cat.fn stdlib/*.fn stdlib/
    funkycmd/*.fn
hello, cat!
hello, cat!
do you cat?
do you cat?
you do cat!
you do cat!
^D
```

At the end, the user pressed the Ctrl+D combination to signal the end of file which caused the program to finish.

We can't do much with just `done`, `putc`, and `getc`, not at least conveniently. That's why we'll now show how to define more complex functions on top of the basic ones to get a more powerful – and sometimes surprisingly powerful – behavior.

3.3 `print`, `println`, `scanln`

The first function with a more complex behavior that we're going to tackle is `print`. The `print` function prints a whole string instead of a single character as `putc` does (it doesn't *do it*, but it instructs the interpreter to do it, and similarly, `print` instructs the interpreter to print a string).

```
func print : String -> IO -> IO =
    \s \next
    if (empty? s)
        next;
    putc (first s);
    print (rest s);
    next
```

The `print` function takes a string and a continuation – the rest of the program. Then it recursively describes how to print the string – print the first character and then continue printing the rest until we printed the whole string. Alternatively, `print` can be defined with a right fold:

```
func print : String -> IO -> IO =
    \s \next fold< putc next s
```

Now we can use `print` to create a convenience `println` function, which additionally prints a newline at the end of the string:

```
func println : String -> IO -> IO =
    print . (++ "\n")
```

The next function we're going to define is `scanln`, which scans a whole line from the input (excluding the newline character) and passes its content. While `print` and `println` prepended some `putc` nodes to the continuation, `scanln` prepends some `getc` nodes, accumulates the line and passes it to the continuation, which accepts one argument: the line string. It's a trailing lambda kind vertical function.

```
func scanln : (String -> IO) -> IO =
    \f
    "" |> fix \loop \s
        getc \c
        if (c == '\n')
            (f (reverse s));
        loop (c :: s)
```

The body makes use of the `fix` function (fix-point operator, `fix f = f (fix f)`) to insert inline recursion. This is a common pattern in Funky. It's used to avoid creating unnecessary helper functions in places where the recursion needs to remember more arguments than the original function has. The recursion in our case has to remember the accumulated string starting from an empty string. The `|>` function (`x |> f = f x`) passes the empty string as the initial value to the recursion.

With the help of `print`, `println`, and `scanln`, we can make more involved programs. Here's a number guessing game:

```
func main : IO =
    println "Think a number from 1 and 100.";
    100 |> 1 |> fix \loop \min \max
        let ((min + max) / 2) \mid
        print (string mid ++ "? ");
        scanln \response
        if (response == "less")
            (loop min (mid - 1));
        if (response == "more")
            (loop (mid + 1) max);
        if (response == "yes")
            (println "Yay!"; done);
        println "Say one of less/more/yes.";
        loop min max
```

And here's an example running of the program:

```
Think a number from 1 and 100.
50? no
Say one of less/more/yes.
50? nope
Say one of less/more/yes.
50? less
25? more
37? more
43? less
40? more
41? more
42? yes
Yay!
```

3.4 `ungetc`, `skip-whitespace`, `scan`

The `print`, `println`, and `scanln` functions only "prepend" operations to the continuation. But since `IO` is a fully transparent data structure, we can define transformations that penetrate it, twist it around, or transform it in any other way.

The first and a very useful example of such a function is `ungetc`. It is used to "push a character back on the input", so that the next `getc` call will get it. The plain `IO` data structure has no such functionality and we can't get any similar behavior just by sequencing `done`, `putc`, and `getc`. What we need is we need to write a function that takes a character and a continuation, then searches through the continuation until it finds the first `getc` node, and finally passes the character to that node. Since `IO` is just a transparent data structure, this is quite easy:

```
func ungetc : Char -> IO -> IO =
    \c \io
    switch io
    case done
        done
    case putc \d \jo
        putc d;
        ungetc c;
        jo
    case getc \f
        f c
```

The `ungetc` function examines the top node of the continuation and recursively propagates itself down the data structure until it finds a `getc` node. When it does, it passes the character to the function of the `getc` node and turns it into its result.

The `ungetc` function is particularly useful for implementing the `scan` function. We've already implemented `scanln`, which scans whole lines. The `scan` function, on the other hand, scans the next full word (a continuous sequence of characters not containing any whitespace) on the input. To do that it first needs to skip all the whitespace preceding the word, then scan the word, but avoid scanning the first whitespace character after the word. We'll see how `ungetc` comes to help with this.

First, we'll make a general function for skipping whitespace on the input:

```
func whitespace? : Char -> Bool =
    \c
    any (c ==) [' ', '\t', '\n', '\r']

func skip-whitespace : IO -> IO =
    \next
    getc \c
    if (whitespace? c) (
        skip-whitespace;
        next
    );
    ungetc c;
    next
```

The `skip-whitespace` function continuously reads characters from the input until it reaches a non-whitespace character. It wasn't supposed to read this character, but it needed to in order to determine whether to stop skipping or not. So it uses `ungetc` to put it back on the input.

With the help of `skip-whitespace`, here's `scan`:

```
func scan : (String -> IO) -> IO =
    \f
    skip-whitespace;
    "" |> fix \loop \s
        getc \c
        if (whitespace? c) (
            ungetc c;
            f (reverse s)
        );
        loop (c :: s)
```

It uses the `skip-whitespace` at the beginning, then it enters a loop where it accumulates the word until it reaches a whitespace again. This whitespace wasn't supposed to be read by `scan`, so it's put back on the input by `ungetc`.

Here's a simple calculator program demonstrating the `scan` function:

```
func main : IO =
    print "> ";
    scan \x-str
    scan \op
    scan \y-str
    # float x-str returns Maybe Float
    # hence the call to extract
    let (extract (float x-str)) \x
    let (extract (float y-str)) \y
    println (
        if (op == "*") (string (x * y));
        if (op == "/") (string (x / y));
        "invalid operation: " ++ op
    );
    main
```

And here's its running:

```
> 10 / 3
3.3333333333333335
> ^D
```

3.5 `reverse-lines`

The last example in our exploring of the possibilities of the `IO` data structure is a rather peculiar one: not practically useful, but quite showing of the potential present here.

The function is called `reverse-lines` and its job is to reverse all the lines on the output. Lines come to the output in two ways: either a sequence of `putc` nodes terminated by a `putc` of a newline, or a sequence of `putc` nodes terminated by a `getc` node – all the pending output must be shown to the user before requesting input and the user input will be entered by a newline.

```
func reverse-lines : IO -> IO =
    \io
    io |> "" |> fix \loop \s \(io : IO)
        switch io
        case done
            done
        case putc \c \jo
            if (c == '\n') (
                println s;
                loop "";
                jo
            );
            loop (c :: s);
            jo
        case getc \f
            print s;
            getc \c
            loop "";
            f c
```

The `reverse-lines` function starts a loop using inline recursion with `fix` to accumulate the line to be reversed. It removes the original `putc` nodes from the data structure and keeps accumulating until reaching one of the above-mentioned conditions for terminating a line. When one of the conditions occurs, it transforms the accumulated reversed line into a proper series of `putc` calls using `print` or `println`.

Here's the small program augmented by `reverse-lines`:

```
func main : IO =
    reverse-lines;
    print " What's your name? ";
    scan \name
    println ("Hello, " ++ name ++ "!");
    done
```

And here's its running:

```
 ?eman ruoy s'tahW
!lahciM ,olleH
```

Unimportant to this paper, this `reverse-lines` function isn't perfect. For example, it fails to reverse the lines of the "cat" program, because that one has a `getc` before each `putc` and so no full line ever gets accumulated. The solution to this problem is left as an exercise to the reader.

4 Conclusion

Combining the concept of vertical code with the new approach to side-effects in a language that makes it viable resulted in a whole new approach to structuring purely functional code. We saw that purely functional programs can be expressed in a way that is familiar to an imperative programmer. In Funky, this doesn't come from artificially implanted syntactic features, but instead stems naturally from the core, general concepts in the language itself.

At the moment, Funky is virtually unknown. We will make all the efforts to get the word out there, because we believe we've got something worthy in our hands.

S. Krajči (ed.): ITAT 2018 Proceedings, pp. 28–34
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Matej Gallo, Luboš Popelínský, and Karel Vaculík

To text summarization by dynamic graph mining

Matej Gallo, Luboš Popelínský, and Karel Vaculík

KD Lab FI MU Brno
popel@fi.muni.cz

Abstract: We show that frequent patterns can contribute to the quality of text summarization. Here we focus on single-document extractive summarization in English. Performance of the frequent patterns based model obtained with DGRMiner yields the most relevant sentences of all compared methods. Two out of three proposed methods outperform other methods if compared on ROUGE data.

1 Introduction

Extractive summarization assigns a score to each text unit (phrase, sentence, paragraph, passage) and then picks the most informative ones to form a summary called extract. The selected sentences are used verbatim [1, 25].

Graph representation is a common way for representing data in automatic text summarization tasks. We introduce a new method for single-document extractive summarization in English language where a text is represented as a dynamic graph and each sentence corresponds to a dynamic graph snapshot, i.e. the graph in a partivular time. E.g. the first sentence is the oldest snapshot while the last sentence of a text is the newest one. This method is based on the principle of mining frequent patterns from such a dynamic graph and using the resulting patterns as indicators of sentence importance.

The structure of this text is following. In Section 2. we introduce DGRMiner, a tool for mining in dynamic graphs. Section 3. describes the algorithm. In Sections 4 we briefly introduce the data used for evaluation. Sections 5 and 6 explain a way of summarization evaluation and sentence scoring. Section 7 displays the main results of text summarization by means of frequent patterns mined from dynamic graphs. Related work is a contents of Section 8. We conclude with concluding Section 9.

2 DGRMiner

We used DGRMiner tool to extract these patterns from the dynamic graphs. DGRMiner was proposed in [26] for mining frequent patterns that capture various changes in dynamic graphs in the form of predictive rules. The found predictive graph rules express what way the graph changes. The rules capture patterns such as addition, deletion, and transformation of the subgraph. The use of relative timestamps for rules allows for mining general patterns while including time information simultaneously. To ensure that only significant rules are extracted, the algorithm incorporates measuring support and confidence. While support expresses what portion of the graph is affected by the rule, confidence measures the occurrence frequency of a specific change, given that a particular pattern was observed. Time abstraction is an extension to the DGRMiner that allows us to analyse broader class of dynamic graphs. The abstraction lies in the use of signum function on relative timestamps. All the negative timestamps become -1, all the positive timestamps become 1 and the timestamp of 0 remains 0. DGRMiner allows for two types of abstraction. One affects only timestamps of vertices and should be used when most changes are caused by edges and vertices remain static. The second one also affects the edges and is useful when there are too few patterns with exact timestamps.

3 Method

The initial idea was to take a text as a temporal (i.e. dynamic) graph where each sentence represent a graph snapshot at a particular time and tokens (a lemma together with a part-of-speech tag) were nodes and edges connected all pairs of tokens. The label of an edge was equal to a number of appearances of this pair of tokens. In the following subsections we describe particular steps of the algorithm.

3.1 Transforming Text to graphs

For pre-processing we employed CoreNLP [15]. CoreNLP provides a set of natural language analysis tools: POS tagger, named entity recognition, parser, coreference resolution system, sentiment analysis, bootstrapped pattern learning, and open information extraction. We used four annotators: sentence split, tokenize, lemma, and POS.

Using the coreNLP package for R, we split the text into sentences. In every iteration we removed the stop words from a sentence, lemmatize the remaining words and assigned the POS tags. If a lemma-tag pair was not already in the graph, we added it in and created edges between all words in a same sentence. Otherwise, we only notified the DGRMiner that a particular word had appeared again. We parametrized context c. Therefore, in each step we modelled at most c sentences.

We will show it on simple example that contains a single sentence.

```
<sentence position = " 9 " labelers
```

```
= "1, 2 , 3, 4">The great thing about
Aspen is that it has at least one of
each of the really useful stores that
you need, sellingstuff at regular
prices. </sentence>
```

After removing stop words and labeling words with a lemma-tag pair, we receive a graph `t # 9`.

```
t # 9
an 74 great#ADJ
an 75 thing#NOUN
an 11 Aspen#NOUN
an 76 least#ADJ
an 77 one#NUM
an 78 really#ADV
an 79 useful#ADJ
an 66 store#NOUN
an 5 need#VERB
ae u 48 5 11 [need#VERB]-[Aspen#NOUN]
ae u 432 5 66 [need#VERB]-[store#NOUN]
ae u 433 5 67 [need#VERB]-[sell#VERB]
ae u 434 5 74 [need#VERB]-[great#ADJ]
ae u 435 5 75 [need#VERB]-[thing#NOUN]
ae u 436 5 76 [need#VERB]-[least#ADJ]
ae u 437 5 77 [need#VERB]-[one#NUM]
```

In the first column, `an, ae` stand for "add a node" and "add an edge" respectively.

3.2 Pattern Mining

Then we applied the DGRMiner on the data from previous step to obtain predictive patterns, i.e. rules that describe frequent (or rare) changes between past and future graphs – sentences. We received two types of patterns: frequent single-vertex patterns corresponding to nodes and rare patterns corresponding to edges. We modified the support parameter to change the threshold on frequency of observed patterns. When the support parameter was set too low, the DGRMiner considered a pattern anything that appeared at least once in the text – every word, every possible word combination. By setting the confidence parameter in DGRMiner to 0, the DGRMiner assigned confidence (as discussed in section 2) to each extracted pattern. We also played with time abstraction which allowed us to ignore the preset context c as discussed in section 2. Therefore, patterns were observed in sentences that were more than $c-2$ sentences apart. Although this gave us more patterns, many of them were uninformative. An example of observed single-vertex pattern is "+supplies#NOUN". This pattern indicates that the noun *supplies* appeared at least n times in the text, where the n is determined by the support parameter. An example of multi-vertex pattern is [store#NOUN]-[sell#VERB], which tells us that the noun *store* is frequently closely accompanied by the verb *sell*. The proximity of these two words is given by context c and the frequency is given by the support parameter.

4 Data

To assess the performance of our method we used the Blog summarization dataset [10, 11, 23]. It consists of 100 posts annotated in XML that were randomly chosen from two blogs (half from each blog), Cosmic Variance and Internet Explorer Blog. Each of the four human summarizers picked approximately 7 sentences to form 4 reference summaries in total. We manually restored apostrophes for shortened forms of to be and to have verbs, and in possessive nouns. Punctuation within sentences was omitted as the coreNLP sentence split annotator often wrongly split the sentences in the middle. We decided not to use CNN Dataset, nor SUMMAC dataset mentioned in [9] because the provided reference summaries were not extracted, verbatim sentences. This would require us to manually find the best matching sentence from within the document.

5 Semi-automatic evaluation

Evaluating summaries on sentence level can be done semi-automatically by measuring content overlap with precision, recall, and F1 measure. An extracted sentence is considered acceptable if the same sentence was extracted in a reference summary. This process cannot be fully automatized because reference summaries are created by human judges. Other semi-automatic evaluation methods used nowadays are: ROUGE, PYRAMID and BASIC ELEMENTS. We will discuss only the ROUGE method [25, 14] as it was used in this work.

5.1 ROUGE

The ROUGE (Recall-Oriented Understudy for Gisting Evaluation) measure is based on the BLEU metrics used in machine translation tasks. The idea is to compare the differences between the distribution of words in the candidate summary and the distribution of words in the reference summaries. Given h reference summaries and a candidate summary they are split into n-grams to calculate the intersection of n-grams between the references and the candidate. This process is illustrated in figure 1.

Given its correlation with manual judgments ROUGE almost seems to have become a standard. [13] reports Pearson coefficient for the most commonly used variations (ROUGE-2 and ROUGE-SU4 [14]) at a value of between 0.94 and 0.99. Generally, the ROUGE-n is calculated from the co-occurrences of n-grams between the candidate and reference summaries as shown by formula 1 in [25]:

$$\text{ROUGE-}n = \frac{\sum_{n\text{-grams}} \in \{\text{Sum}_{\text{can}} \cap \text{Sum}_{\text{ref}}\}}{\sum_{n\text{-grams}} \in \text{Sum}_{\text{ref}}} \qquad (1)$$

where the numerator is the maximum number of co-occurrences of n-grams in both reference and candidate

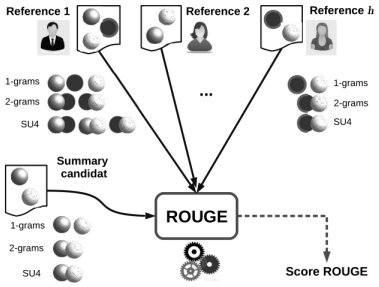


Figure 1: The basic idea of ROUGE for evaluation of summaries [25]

summary and the denominator is the total sum of the number of n-grams present in the reference summaries.

The ROUGE-SUγ is an adaptation of ROUGE-2 using skip units (SU) of a size $\leq \gamma$. SU4 considers the bigrams and the bigrams SU to be arbitrary in a maximum window length of $\gamma = 4$ words. The number of bigrams with an arbitrary size of length γ is given by formula 2:

$$\text{Count}(k,n) = C\binom{n}{k} - \sum_{0}^{k-\gamma}(k-\gamma); \gamma \leq 1 \qquad (2)$$

where n is the n-gram length and k is the sentence length in words.

The ROUGE metrics are not flawless. Firstly, they have a problem with the representation of content. Secondly, they will not consider chains of words such as "MUNI" $\neq$ "Masaryk University" and "FI" $\neq$" Faculty of Informatics". A study [22] found that the system can be tricked into generating a summary with high ROUGE score.

6 Sentence Scoring

In this step we used the observed patterns as indicators to score the sentences. The final score was obtained according to formula 3 as a sum of single-vertex and multi-vertex scores:

$$\text{Score}(s) = \text{Score}_{single}(s) + \text{Score}_{multi}(s) \qquad (3)$$

Three different metrics were implemented to calculate single-vertex score.

The Jaccard coefficient (**JAC**) as given by formula 4, expresses the overlap of two sets. In our case, between the set of patterns P and the set of words in the sentence S.

$$\text{Score}_J(s) = \frac{w_{sum}(S \cap P)}{w_{sum}(S \cup P)} \qquad (4)$$

where w_{sum} assigns weight to every element of the set and then sums over them. A question arises regarding what weight we should assign to non-indicators. Empirically, we received the best results for the value of 0.001.

The frequency method (**FRQ**) as given by formula 5, is the simplest of the three. For a sentence s and a set of patterns P the score is calculated as weighted average.

$$\text{Score}_F(s) = \sum_{p \in P} c(p)w(p) \qquad (5)$$

where $c(p)$ is the frequency of the pattern in the sentence and $w(p)$ is its associated weight.

The density method (**DEN**) as given by formula 6, counts the patterns in a sentence and normalizes by the length of the sentence. This method is parameter-free.

$$\text{Score}_D(s) = \frac{|\, S \cap P \,|}{\text{length}(s)} \qquad (6)$$

For multi-vertex patterns we tested frequency and density methods. The only difference between the methods is that instead of length of sentences we use the number of all possible word pair combinations $\binom{|S|}{2}$.

We built the summary in a greedy fashion. In every iteration we picked the highest scoring sentence. Patterns observed in the sentence were penalized by a parameter λ. The scores were recomputed and the process continued until a desired number of sentences was picked or until all the sentences were used. For every method we discovered the optimal value of λ. We tuned the parameter on 10% of the entire dataset. The optimal values for λ are presented in the tables 2 and 1. The ordering of sentences in the final summary maintains the relative ordering in the original document.

7 Results

We evaluated the model using the ROUGE-1 metric, which is recommended for short summaries [14]. Every blog post was compared against four reference summaries as described in chapter 4. The results of all three methods can be seen in tables 1, 2 and 3. The first three columns denote whether time abstraction on both vertices and edges is used, whether the patterns were used to score sentences, and whether the initial weights were determined by the confidence of DGRMiner for the given pattern, respectively. The last three columns correspond to the ROUGE-1 metrics – precision (what portion of sentences we selected were part of the reference summary), recall (what portion of sentences in reference summary we extracted), and f1 measure (harmonic mean of precision and recall).

The FRQ method marked the most accurate sentences among all the presented algorithms as can be seen from figure 2. JAC method picked fewer correct sentences than FRQ method but still more than any traditional approach. Both FRQ and DEN methods ranked first in terms of precision.

The frequency-based method incorporating patterns with no confidence weighting (identified as FRQ) achieved the highest recall, the highest f1 score, and the highest

binall	patterns	weights	precision	recall	f1 score
F	F	F	0.643	0.694	0.664
F	T	F	0.643	0.686	0.659
F	T	T	0.641	0.684	0.657
T	F	F	0.641	0.683	0.657
F	F	T	0.640	0.691	0.660
T	T	F	0.640	0.678	0.654
T	T	T	0.640	0.678	0.654
T	F	T	0.639	0.681	0.655

Table 1: Results for blog summarization dataset using jaccard method with parameters $\lambda = 0.55$, $w_0 = 0.001$

binall	patterns	weights	precision	recall	f1 score
F	T	F	0.645	0.702	0.668
F	F	T	0.645	0.700	0.667
F	F	F	0.644	0.701	0.665
F	T	T	0.643	0.693	0.662
T	F	F	0.642	0.691	0.661
T	T	F	0.642	0.691	0.661
T	F	T	0.642	0.690	0.660
T	T	T	0.642	0.688	0.660

Table 2: Results for blog summarization dataset using frequency method with parameter $\lambda = 0.20$

binall	patterns	precision	recall	f1 score
F	T	0.651	0.514	0.562
F	F	0.650	0.514	0.562
T	T	0.649	0.516	0.563
T	F	0.649	0.512	0.562

Table 3: Results for blog summarization dataset using density method

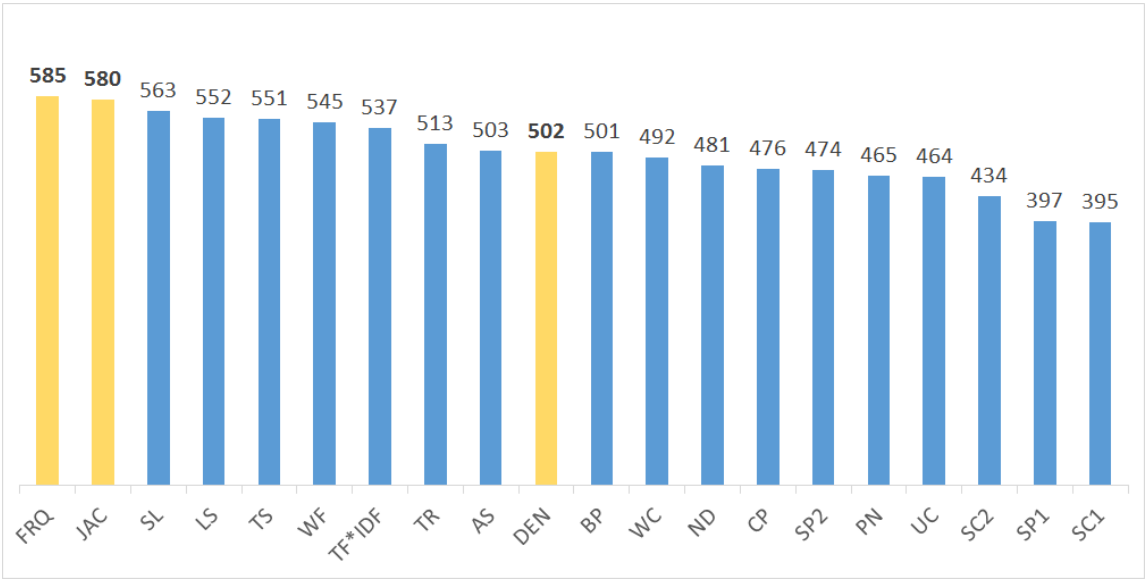


Figure 2: Comparison of number of correctly chosen sentences – using blog summarization dataset (our algorithms are displayed yellow)

number of correctly chosen sentences. The highest precision was achieved by density-based method incorporating patterns with no confidence weighting (identified as DEN). We chose these two models and the highest scoring Jaccard method for comparison together with algorithms presented in article [9]. We could see that FRQ behaves similarly to *Word frequency (WF)* algorithm proposed in the paper. This is not surprising, because the single vertex-frequent patterns correspond to words that appear at least n-times in the text. Where n is determined by the support parameter in the DGRMiner tool as described in section 3.2. The multi-vertex patterns improved the performance of the summarizer in density models and in one case (the highest scoring model) in frequency model.

8 Related work

Kupiec et al. introduce in their work [12] method inspired by Edmundson's [7]. They approached the summarization as a statistical classification problem. A Bayes classifier was trained to estimate the probability that a given sentence would be included in the summary. They used 6 discrete features (presented in order of importance): paragraph feature (the position of sentence in paragraph s), fixed-phrase feature (the sentence contains a phrase from a list), sentence length cutoff feature (threshold $u_1 = 5$, length$(s) > u_1$), thematic word feature (the presence of thematic terms), uppercase word feature (the presence of words in capital letters). The best results were obtained using the first three features.

Aone et al. [2] built on Kupiec's work [12] and expanded the feature set of their system, called DIMSUM, with signature terms, which indicate key concepts for a given document. Another advantage over [12]'s system is the use of multi-word phrases – statistically derived collocation phrases (e.g. "omnibus bill", "crime bill", "brady bill") and associated words ("camera" and "obscura", "Columbia River" and "gorge"), as well as the use of WordNet [16] to identify possible synonyms of found signature terms. He applied a shallow discourse analysis to resolve co-references and maintain cohesion – only name aliases were resolved such as UK to United Kingdom.

Osborne in his work [18] disagrees with the traditional assumption of feature independence and shows empirically that the maximum entropy (MaxEnt) model produces better extracts than the naïve Bayes model with similarly optimized prior appended to both models. Unlike naïve Bayes, MaxEnt does not make unnecessary feature independence assumptions. Let c be a binary label (binary: part of summary or not), s the item we are interested in labeling, f_i the i-th feature, and ω_i the corresponding feature weight.

Hidden Markov Models (HMM), similar to MaxEnt, have weaker assumptions of independence. There are three types of dependencies: positional dependence, feature dependence, and Markovity dependence. A first-order Markov model allows modeling these dependencies. Conroy and O'leary [6] use a joint distribution for the features set, unlike the independence-of-features assumption used in naïve Bayesian methods. The HMM was trained using five features: position of the sentence in the document (number of states); number of terms in a sentence, and likeliness of the sentence terms given the document terms.

Summarizer NetSum presented by Svore et al. [24] uses an artificial neural network (ANN) called RankNet to rank the sentences. RankNet is a pair-based neural network algorithm for ranking a set of inputs. It is trained on pairs of sentences (S_i, S_j), such that the ROUGE score for S_i should be higher than S_j. Pairs are only generated in a single document, not across documents. The cost function for RankNet is the probabilistic cross-entropy cost function. Training is performed using a modified version of back-propagation algorithm for two-layer networks, which is based on optimizing the cost function by gradient descent. The system significantly outperforms the standard baseline in the ROUGE-1 measure. No past system could outperform the baseline with statistical significance.

A system for generating product category-based (topic-based) extractive summarization was proposed by [4, 5]. The collection of 45 news items corresponding to various products are pre-processed using standard techniques: tokenization, stopword removal, stemming. The final corpus contains around 1500 features and is represented by a bag-of-words VSM based on these features. To identify the topics, the news items about specific categories of products are segregated into separate clusters using K-Means and then an extractive summary is generated from each of these topical clusters. The K number of cluster is determined by a Self-Organizing Map (SOM).

Chakraborti and Dey [5] assigned a score to the entire summary as a single unit. The total summary score (TSS) is taken as a combination of cosine similarity between centroid of corpus and the summary as a whole; relative length of the summary; and redundancy penalty. To maximize TSS constrained by the number of lines in summary ($\tau = 5-7\%$), they opted for quick Artificial Bee Colony optimization, a global optimization technique, for sentence selection. The summary with the highest score is then chosen.

In Muresan's paper [17], a system called GIST-IT used for email-summarization task is discussed. First, noun phrases (NPs) are extracted as they carry the most contentful information. Subsequently, machine learning is used to select the most salient NPs. A set of nine features, divided into three categories (head of the NP, whole NP, combination of head and modifiers of NP) were used: head of the NP TF*IDF, position of first occurrence (focc) of the head in text, TF*IDF of entire NP, focc of entire NP, length of the NP in words, length of the NP in characters, position of the NP in the sentence, position of the NP in the paragraph, and combination of the TF*IDF scores of head of the NP and its modifiers.

To find the most salient NPs, three machine learning al-

gorithms were applied: decision trees (axis-parallel trees – C4.5 and oblique trees – OC1), rule induction (production rule – C4.5rules and propositional rules – RIPPER), and decision forests (DFC using information gain ratio).

Muresan claims that shallow linguistic filtering applied to NPs improved the classifiers accuracy [17]. The filtering consisted of four steps: grouping inflectional variants, removing unimportant modifiers, filtering stopwords, and removing empty nouns.

A modification of PageRank algorithm called LexRank is used to weight sentences. The undirected graph of sentences is constructed from symmetrical similarities (modified cosine measure). The score of each vertex s is calculated iteratively until the values of the vertices have not been modified by more than $\varepsilon = 0.0001$. LexRank algorithm is used as a component of the MEAD [8].

Unlike LexRank, TextRank uses the similarities of edges to weight the vertices. The score of each sentence s_i is calculated iteratively until convergence is reached.

As the graph is constructed from inter-sentence similarity measure, the choice of the method for sentence-weighting has significant impact. One approach is to use a bag-of-words to represent the sentences. The similarity is obtained by calculating the cosine similarity weighted by inverse document frequency [1] between their vectorial representations. Another approach suggests using word overlap between sentences instead. The weak point of all similarity measures that use words (cosine similarity, word overlap, longest common subsequence) is the dependency on the lexicon of the document. The solution to this is its combining with similarity measure based on chains of characters. Therefore, sentences that do not share a single word but contain a number of words that are close morphologically can be compared [25].

Patil [19] proposed a new graph-based model called SUMGRAPH. First the text is pre-processed (stemmed) and represented as a VSM with TF*IDF weights. He then computes pair-wise cosine similarities and subtracts the values from 1 to obtain dissimilarities. The resulting matrix of intra-sentence dissimilarities is then used to model the document as graph. The vertices represent the sentences and edges are weighted by intra-sentence dissimilarities.

The novel idea is the use of link reduction technique known as Pathfinder Network Scaling (PFnet) [21, 20] to scale the graph. PFnet models the human aspects of semantic memory. The centrality of a sentence and its position in a document are used to compute the importance of sentences. Four different centrality measure were tested and closeness centrality showed to perform best. Finally, the sentences are ranked according to their importance and first n highest-scoring sentences were picked.

The approaches based on similarity graphs solely model the similarity between pairs of sentences with no clear representation of word relations. Therefore, it is not clear if they adequately cover all topical information. The hypergraph-based approach is to remedy this problem by capturing the high-order relations between both sentences and words. [3] proposed a hypergraph-based model for generic summarization based on [27]'s hypergraph-based model for query-focused summarization. The ranking uses a semi-supervised approach to order sentences. They model the words as vertices and sentences as hyperedges and then approach the problem as a random walk over hyperedges.

9 Conclusion

We showed that frequent patterns can contribute to the quality of text summarization. The comparison supports our statement that the performance of our frequent patterns based model is comparable to the simpler word frequency method and yielded the most relevant sentences of all compared methods. Our methods outperformed other methods in precision but lacked in recall. We attribute the similarity to word frequency method to inadequate graph representation – instead of interconnecting all the words within a sentence, suggest connecting them according to the parse tree. Another consideration is to use a graph mining tool that searches for more specific types of patterns than DGR-Miner.

Acknowledgments

This work has been supported by Faculty of Informatics, Masaryk University. We would like to thank to ITAT reviewers for their suggestions and comments.

References

[1] Charu C. AGGARWAL and ChengXiang ZHAI. *Mining Text Data*. Springer, New York, 2012.

[2] Chinatsu AONE, James GORLINSKY, and Bjornar LARSEN. A trainable summarizer with knowledge acquired from robust nlp techniques. *Advances in Automatic Text Summarization*, 71, 1999.

[3] Abdelghani BELLAACHIA and Mohammed AL-DHELAAN. Multi-document hyperedge-based ranking for text summarization. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1919–1922, New York, 2014 [cit. 2016-05-10]. ACM.

[4] Swapnajit CHAKRABORTI. Multi-document text summarization for competitor intelligence: A methodology based on topic identification and artificial bee colony optimization. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 1110–1111, New York, 2015 [cit. 2016-05-10]. ACM.

[5] Swapnajit CHAKRABORTI and Shubhamoy DEY. Product news summarization for competitor intelligence using topic identification and artificial bee colony optimization. In *Proceedings of the 2015 Conference on Research in Adaptive and Convergent Systems*, pages 1–6, New York, 2015 [cit. 2016-05-10]. ACM.

[6] John M. CONROY and Dianne P. O'LEARY. Text summarization via hidden markov models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 406–407, New York, 2001 [cit. 2016-05-10]. ACM.

[7] Harold P. EDMUNDSON. New methods in automatic extracting. *J. ACM [online]*, 16(2):264–285, april 1969 [cit. 2016-05-10].

[8] Günes ERKAN and Dragomir R. RADEV. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res. [online]*, 22(1):457–479, december 2004 [cit. 2016-05-11].

[9] Rafael FERREIRA, Luciano CABRAL, and Rafael D. LINS. Assessing sentence scoring techniques for extractive text summarization. *Expert Systems with Applications [online]*, 40(14):5755 – 5764, october 2013 [cit. 2016-05-10].

[10] Meishan HU, Aixin SUN, and Ee-Peng LIM. Comments-oriented blog summarization by sentence extraction. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, pages 901–904, New York, 2007 [cit. 2016-05-10]. ACM.

[11] Meishan HU, Aixin SUN, and Ee-Peng LIM. Comments-oriented document summarization: Understanding documents with readers' feedback. In Sung-Hyon MYAENG, Douglas W. OARD, Fabrizio SEBASTIANI, Tat-Seng CHUA, and Mun-Kew LEONG, editors, *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 291–298, New York, 2008 [cit. 2016-05-10]. ACM.

[12] Julian KUPIEC, Jan PEDERSEN, and Francine CHEN. A trainable document summarizer. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 68–73, New York, 1995 [cit. 2016-05-10]. ACM.

[13] Chin-Yew LIN. Rouge: a package for automatic evaluation of summaries. In Stan SZPAKOWICZ Marie-Francine MOENS, editor, *Workshop Text summarization Branches Out (ACL '04) [online]*, pages 74–81, Barcelona, 2004 [cit. 2016-05-10]. ACL.

[14] Petr MACHOVEC. Automatická sumarizace textu [online]. Master's thesis, Masaryk University, Faculty of Informatics, Brno, 2015 [cit. 2016-05-10].

[15] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014.

[16] George A. MILLER. Wordnet: A lexical database for english. *Commun. ACM [online]*, 38(11):39–41, november 1995 [cit. 2016-05-12].

[17] Smaranda MURESAN, Evelyne TZOUKERMANN, and Judith L. KLAVANS. Combining linguistic and machine learning techniques for email summarization. In *Proceedings of the 2001 Workshop on Computational Natural Language Learning - Volume 7*, pages 19:1–19:8, Stroudsburg, 2001 [cit. 2016-05-10]. Association for Computational Linguistics.

[18] Miles OSBORNE. Using maximum entropy for sentence extraction. In *Proceedings of the ACL'02 Workshop on Automatic Summarization*, Stroudsburg, 2002 [cit. 2016-05-10]. Association for Computational Linguistics.

[19] Kaustubh PATIL and Pavel BRAZDIL. Text summarization: Using centrality in the pathfinder network. *Int. J. Comput. Sci. Inform. Syst [online]*, 2:18–32, 2007 [cit. 2016-05-12].

[20] Roger W. SCHVANEVELDT, editor. *Pathfinder Associative Networks: Studies in Knowledge Organization*. Ablex Publishing Corp., Norwood, 1990.

[21] Roger W. SCHVANEVELDT, D.W. DEARHOLT, and F.T. DURSO. Graph theoretic foundations of pathfinder networks. *Computers & mathematics with applications [online]*, 15(4):337–345, 1988 [cit. 2016-05-11].

[22] Jonas SJÖBERGH. Older versions of the rougeeval summarization evaluation system were easier to fool. *Inf. Process. Manage.*, 43(6):1500–1505, november 2007 [cit. 2016-05-12].

[23] Dr. Aixin SUN. Comments-oriented document summarization, 2015. http://www.ntu.edu.sg/home/axsun/datasets.html, visited 2016-05-20.

[24] Krysta M. SVORE, Lucy VANDERWENDE, and Christopher J.C. BURGES. Enhancing single-document summarization by combining RankNet and third-party sources. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 448–457, Prague, 2007 [cit. 2016-05-10]. Association for Computational Linguistics.

[25] Juan-Manuel TORRES-MORENO. *Automatic Text Summarization*. ISTE, London, 2014.

[26] Karel Vaculík and Lubomír Popelínský. Dgrminer: Anomaly detection and explanation in dynamic graphs. In Knobbe-A. Soares C. Papapetrou P. Boström, H., editor, *Advances in Intelligent Data Analysis XV - 15th International Symposium, IDA 2016*, pages 308–319, Neuveden, 2016. Springer.

[27] Wei WANG, Furu WEI, Wenjie LI, and Sujian LI. Hypersum: Hypergraph based semi-supervised sentence ranking for query-oriented summarization. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pages 1855–1858, New York, 2009 [cit. 2016-05-10]. ACM.

S. Krajči (ed.): ITAT 2018 Proceedings, pp. 35–40
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Oldřich Krůza

Phonetic Transcription by Untrained Annotators

Oldřich Krůza

Charles University, Faculty of Mathematics and Physics,
Institute of Formal and Applied Linguistics
kruza@ufal.mff.cuni.cz

Abstract: The paper presents an application for lay, untrained users to generate high-quality, aligned phonetic transcription of speech. The application has been in use for several years and has served to transcribe over 600 thousand word forms over two versions of a web interface. We present measures for compensating the lack of expert training.

1 Introduction

1.1 Our Setting

The work presented in this paper is a part of the project that tends the spoken corpus of Karel Makoň[1]. The corpus is of the single speaker and has been recorded in amateur conditions, while the author was speaking to his friends about a novel way to interpret the teaching of Jesus and of mystic and spirituality in general. Karel Makoň died in 1993 and a community of favorers of his teachings has persevered since then.

The talks can be seen as companions to Makoň's written works. Together they form a unique, extensive, consistent systematization of the spiritual path tailored to modern westerners and accessible primarily to Czech speakers. It draws heavily on traditional Christian mysticism as well as ancient tradition of India and China, adapting them for the present. The whole system can be seen as a manual for entering the eternal life prior to the physical death.

There are over 1000 hours of digitized recordings of Karel Makoň, they are accessible under the CC-BY license and the project aims at bringing the most benefit out of them. The first step was digitizing the recordings from the original magnetic tapes, the second step was releasing all of them on the world-wide web, the third step was developing a web-based system for human / machine transcription of the bulk, allowing for search.

The transcription we do is both phonetic and orthographic.[1] Our users are supposed provide orthographic transcription where the pronunciation is standard and phonetic otherwise.

[1]There is no actual focus on orthography. Instead, we mean the natural way of transcribing the speech to human-readable text. Where it matters, focus is directed at precise correspondence with the utterances instead of language cleanliness.

1.2 Architecture Overview

The system consists of

1. The corpus in compressed audio format. We use mp3 and ogg/vorbis to accomodate most browsers. These data are hosted on an external CDN.
2. The exact copy of the corpus in parametrized (MFCC) format. These data reside on the back-end server.
3. A complete, aligned transcription of the recordings, hosted on the back-end server and mirrored on a CDN.
4. Acoustic model trained on the human-transcribed part of the corpus.
5. Language model trained using Srilm[2] on a combination of publicly available Czech texts, Karel Makoň's written works, and both the human-submitted and automatically-acquired transcription.
6. Back-end API for collecting correcions to the transcription, serving the transcription and allowing full-text search with elasticsearch[2].
7. Separately hosted front-end web application serving as an interface for playing the recordings, synchronously displaying the transcriptions and collecting the corrections from users.

To get the initial transcription, we have manually transcribed some 10 minutes of the material using Transcriber[3], trained an acoustic model on it and recognized the whole data using it.

2 Annotator Expertise

Our case is on the edge of what can be called linguistic data annotation. In our lucky part of the world where alphabetization nears 100%, transcription of speech is hardly expert work. On the other hand, ensuring that the transcription exactly matches the audio

[2]https://www.elastic.co/products/elasticsearch
[3]http://trans.sourceforge.net/

- as a representation of the words uttered and of their meaning,
- on the phonetic level, phone for phoneme,[4]
- on the time axis

is beyond what can be expected from an untrained user.

Linguistic data annotation in general requires trained personnel. If we only look at the Prague Dependency Treebank, we can notice the annotators provided such a degree of expertise they have become the co-authors[3].

Crowdsourcing, community-driven approach or engaging volunteers is an ever stronger, popular way of obtaining assets that would otherwise be unbearably costly. Let us mention for example Mihalcea (2004)[4] who delegates word-sense disambiguation to volunteers. The Wikicorpus[5] as well as the MASC[6] gather annotation from volunteers.

In most cases, quality is very important for data annotation, so some kind of control is essential, no matter how expert the annotators. Trivially, the less expertise, the more control is needed.

2.1 Quality Control

A common way of dealing with quality control is to inspect annotator agreement. This has the huge downside that every piece of data must be annotated at least twice, which reduces the yield by 50+%.

There is another reason not to use it in our case. Our application is designed for people who want to listen to the recordings out of interest and their contribution to the quality of the transcription is more of a by-product. It would be hard to convince them to choose exactly a recording that another user has already transcribed.

Luckily, we can implement automatic measures to aid the annotators to deliver higher-quality transcription.

2.2 Forced Alignment

We always assume an existing transcription, so we can see the user's contribution as a correction. Every submission has the form of replacing a text segment with another. Since the transcriptions are time-aligned to the audio, we also know exactly what is the corresponding audio segment to the text submitted.

This enables us to perform forced alignment on the submitted text and the audio. With a well selected pruning threshold, we can distinguish false transcriptions and reject them, providing feedback to the contibutor. Since every segment of audio fits the acoustic model to a different degree, both false positives and false negatives will inevitably occur.

False positives (when the system accepts a wrong transcription) present a problem, since the error will enter the training data set. But users can often circumvent false negatives by submitting the transcription divided in different segments. Of course, this method can also be used to force a wrong transcription but we assume no malevolence on the part of the users.

Apart from catching wrong transcription, the forced alignment mechanism provides exact synchronization on the time axis. This is a completely missing element in the case of virtually all programs for computer-aided transcription. For some examples, Transcriber, a veteran open-source transcribing program for Linux, expects the user to provide alignment on the level of phrases; Transcribe,[5] a commercial web-based transcribing tool, allows the user to add timestamps anywhere in the text. There is no acoustic model, hence nothing to match against.

3 Phonetic Transcription

3.1 Purpose

We have originally built the acoustic model using HTK,[6] the Hidden Markov Model Toolkit. Here, explicit phonetically labeled training data are necessary for training. We are switching to DNN, using Mozilla's DeepSpeech,[7] where no explicit phonetic annotation is needed but for some purposes like forced alignment, the original HMM is still irreplaceable.

Also, the phonetic labeling is valuable per se for research purposes.

3.2 Phoneme Set

We use a subset of PACal[7]. We shall also refer to individual phonemes in this paper using the PACal notation in `monospace font`. For reference, Table 1 lists the phonemes used with their IPA notation.

3.3 Acquisition

The phonetic transcription is in normal case also a product of forced alignment, as in case of pronunciation variants, it selects the most fitting one. This requires a way to automatically obtain all pronunciation variants of any word. We use a combination of a rule-based system inspired by Psutka et al.[8], in combination with a dynamic dictionary. The dynamic dictionary is a list of alternative pronunciations of a word, which expands as the app is being used.

[4] In the sense that each written phoneme corresponds to exactly one uttered phone.

[5] https://transcribe.wreally.com/

[6] http://htk.eng.cam.ac.uk/

[7] https://github.com/mozilla/DeepSpeech

IPA	PACal	common grapheme	IPA	PACal	common grapheme
a	a	a	ɱ	mg	tra͟mvaj
aː	aa	á	n	n	n͟e
aʊ̯	aw	au	ŋ	ng	tan͟k
b	b	b	ɲ	nj	ň
t͡s	c	c	o	o	o
t͡ʃ	ch	č	oː	oo	ó
d	d	d	oʊ̯	ow	ou
ɟ	dj	ď	p	p	p
d͡z	dz	dz	r	r	r
d͡ʒ	dzh	dž	r̥̊	rsh	t̥r̥i
ɛ	e	e	r̥	rzh	ř̥íz
ɛː	ee	é	s	s	s
eʊ̯	ew	eu	ʃ	sh	š
f	f	f	t	t	t
g	g	g	c	tj	ť
ɦ	h	h	ʊ	u	u
i	i	i	uː	uu	ú, ů
iː	ii	í	v	v	v
j	j	j	x	x	ch
k	k	k	z	z	z
l	l	l	ʒ	zh	ž
m	m	m͟ák		sil	
				sp	

Table 1: Phonemes used in transcription

The users are instructed to transcribe any words with non-standard pronunciation phonetically and then correct their orthographical form. This is one of the few cases where we are coercing the users to something.

When the orthographically broken, phonetic transcription of a word is submitted, if it passes the forced-alignment phase, it is integrated into the displayed transcription. The word's data representation consists of its

1. occurrence: the word as it appears in the text, including capitalization and punctuation,
2. wordform: the word as it appears in the language model and phonetic dictionary (computed as the occurrence in lowercase and stripped of non-alphabetic characters[8]),
3. pronunciation: an array of phonemes,
4. timestamp: distance of the beginning of the word from the beginning of the file, in seconds, in precision of 2 decimal digits,
5. manual/automatic: boolean flag denoting whether the word has been transcribed manually or not,
6. confidence measure: in case of automatically acquired words, the confidence-measure score of the recognizer.

Once merged into the displayed transcription, each word's occurrence can be edited manually. Now the user can enter the correct form deviating from Czech pronunciation rules.

Doing so results in adding the wordform-pronunciation couple to the dynamic pronunciation dictionary and is also used for forced alignment. Thus, this operation need only be performed once per word and any subsequent time the word is entered in its standard orthographic form, the correct pronunciation is inferred.

For example, let's examine the scenario of transcribing the sentence *Proč se toto nestalo Marii Markétě Alacoque? (Why hasn't this happened to Mary Margaret Alacoque?)* Its phonetic representation is `p r o ch sp s e sp t o t o sp n e s t a l o sp m a r i j i sp m a r k ee tj e sp a l a k o k sil .`

1. Suppose the user enters the correct ortographic transcription.
2. The phonetic transducer outputs `p r o ch sp s e sp t o t o sp n e s t a l o sp m a r i j i sp m a r k ee tj e sp a l a c o k v u e sil .`

[8] This implies that all non-alphabetic characters are always a part of a token and never form a token on their own.

3. With a bit of luck, the forced alignment fails because of the distiction of the phone sequence `k o k` and `c o k v u e`.
4. The transcription is rejected, the user realizes that the word is pronounced in a non-standard way and re-tries with *Proč se toto nestalo Marii Markétě alakok?*
5. Forced alignment succeeds now and the entered transcription is merged into the view.
6. The user selects the non-existent word *alakok?* and edits its occurrence to *Alacoque?*
7. Now the word is correctly stored and on any subsequent user inputs of *Alacoque* with any punctuation or capitalization, the pronunciation `a l a k o k` is inferred by the forced alignment.

3.4 Phonetic Respelling

With all advantages of using PACal as a representation for phonemes, it is clearly not the most natural way for lay Czechs to write down and read literal pronunciation. Thanks to the simple, mostly deterministic mapping between phonemes and graphemes, pronunciation respelling is a reliable, natural way. There's not even a need for explicit syllable separation as seen in English pronunciation respelling (wikipedia[9] gives the example *"Diarrhoea" is pronounced DYE-uh-REE-a*). We postulate that the phonetic respelling is natural to all alphabetized native Czech speakers as a fact without any supporting research, based on experience alone.

The previous subsection gave an example of using pronunciation respelling in Czech with the example of *alakok* for *Alacoque*. The direction from the phonetic respelling to the phoneme array is covered by the ortographic-to-phonetic transducer. But we also need the opposite direction to provide the users a way to check whether the pronunciation selected by the forced alignment fits.

For this purpose, we have created a JavaScript module for transduction between the array of phonemes and the pronunciation respelling.[10]

The algorithm is simple. In most cases, a phoneme corresponds uniquely to one character in the respelling. Exceptions are as follows:

1. The phoneme `x` is spelled *ch*.
2. The phonemes `dz dzh` are spelled *dz dž*.
3. The diphtongs `aw ew ow` are spelled *au eu ou*.
4. Sequences `c h, o u, a u, e u, d z, d zh` are spelled *c'h, o'u, a'u, e'u, d'z, d'ž*. Note though, that the sequence `c h` is purely hypothetical, as it contradicts voiced/voiceless assimilation.
5. Voiceless alveolar fricative trill is explicated as *r'*.
6. Palatal nasal and labiodental nasal are spelled *n', m'*.
7. Trailing silence is not represented.

The module includes two-way transduction, although only the one from array of phonemes to human-readable phonetic respelling is needed in our application. Still, the user can mark up special-case pronunciation with the apostrophe, like the sequence of phonemes `o` and `u` with the string `o'u`. The need has never occurred during the six years' lifespan of the application.

Note that when encoding into the phonetic respelling, none of *di ti ni dě tě ně* is ever output. The palatal consonants are always explicitly spelled out and e.g. the sequence `n i` is always spelled *ny*

A few examples of words, pronunciation and phonetic respelling as output by the algorithm (given the corresponding pronunciation is input as phoneme list):

- nic /`nj i c`/: *ňic*,
- kdo /`g d o`/: *gdo*,
- disk /`d i s k`/: *dysk*,
- dřít /`d rzh ii t`/: *dřít*,
- třít /`t rsh ii t`/: *tř'ít*,
- auto /`aw t o`/: *auto*,
- nauka /`n a u k a`/: *na'uka*,
- džbán /`dzh b aa n`/: *džbán*,
- odžít /`o d dz ii t`/: *od'žít*,
- odznak /`o dz n a k`/: *odznak*,
- podzemí /`p o d z e m ii`/: *pod'zemí*,
- noc /`n o c`/: *noc*,
- tento /`t e n t o`/: *tento*,
- hangár /`h a ng g aa r`/: *han'gár*,
- samba /`s a m b a`/: *samba*,
- tonfa /`t o mg f a`/: *tom'fa*.

The use of apostrophe for distinguishing ambiguities and special cases is not 100% intuitive and presents another point where instruction is necessary for the user to use this feature properly.

[9] https://en.wikipedia.org/wiki/Pronunciation_respelling
[10] https://github.com/Sixtease/MakonReact/blob/master/src/lib/Phonet.js

4 Evaluation

We have presented our web application as a tool that enables gathering precisely aligned, phoneme-exact transcription from untrained casual visitors. We have presented measures for reaching this goal but the degree to which it was reached remains unclear.

We have no gold standard data to measure the quality of our manual transcriptions. On the contrary, we use the manual transcriptions as gold standard for the automatic recognition. What we can to, however, is look at some random samples and try to get a rough idea of how the system performs.

4.1 Validation by Forced Alignment

One thing we can examine are the approvals / rejections of the forced alignment. Of 109640 forced alignment attempts, 3419 have failed, which makes for 3.12% rejection rate. We have manually inspected 20 random failed attempts and came to the following numbers:

- 11 cases were false negatives, where the transcription was correct and should have been accepted,
- 4 cases were caused by acoustic irregularities like noise,
- 4 cases were true negatives caused by wrongly chosen segment boundaries and
- 1 case was true negative caused by wrong transcription.

Hence, in 25% of the minimalistic sample, the forced alignment did its job of a validator and prevented a piece of broken training data from entering the dataset. In 55% it was a nuisance and failure, and in the remaining 20%, it rejected a valid transcription but prevented a bad training example from occurring, so we can see this in positive light.

4.2 Non-Standard Pronunciation

We can also track how the scenario described in subsection 3.4 is applied. We have looked up four promising example records in the dynamic dictionary and checked submitted transcriptions containing them. Table 2 lists for each of them the correct orthographic form, the wrong pronunciation obtained by the transducer, the correct pronunciation and finally the phonetic respelling. Each is followed by the number of occurrences in the manually transcribed data.

We can see in Table 3 that the majority of cases results in both orthographic and phonetic forms being correct. Only in about 13% cases, the orthographically incorrect form is kept. We attribute this to the fact that those who use the phonetic respelling are aware of the problematic and mostly go the whole way and clean up.

On the other hand, nearly a third of the cases show the wrong phonetic representation. This is a serious problem on at least two levels: Firstly, it shows that the forced aligner failed to catch the error. Secondly, it lets bad examples into the training dataset.

One of the apparent reasons for this to happen is that the dynamic dictionary only recognizes exact matches. We can see in one file, for example, all occurrences of the form *Weinfurter* to have correct pronunciation while *Weinfurterovi* to have a broken one.

Other factors likely include user carelessness or ignorance, which is exactly what our application is trying to compensate, but fails in these cases.

The cases with false orthographic form don't pose much of a problem. It can harden searching for the term in question but performing a search for the phonetic respelling or even automatically searching the pronunciation would easily mitigate this.

The fourth combination of phonetic respelling and false pronunciation is of course not occuring.

5 Conclusion

We have presented an application that has been providing access to the extensive corpus of Karel Makoň and to acquire an almost complete transcription thereof. Nearly 70 hours corresponding to over 600,000 word forms have been transcribed manually with minimal financial[11] as well as development[12] costs. Only some of the volunteers have indulged instruction time in order of minutes. The rest of the corpus has been transcribed using an ASR system trained on these ever-growing data.

We have presented the ways we use to aid the untrained users to provide a high-quality orthographic and phonetic time-aligned transcription. We have attempted a rough evaluation of the success rate of the measures presented. Though clearly far from perfect, they do serve the purpose and set a baseline for improvements or novel approaches.

The system has been built with the motivation of spreading the message contained in Karel Makoň's talks. However, to make the technology more useful, we are actively looking for similar settings where it could be deployed.

Acknowledgments

The research was supported by SVV project number 260 453.

[11] In early stages, we kept a paid annotator to test the application.

[12] The system has been written by a single developer.

Correct spelling	#	wrong phonetic pronunc.	#	correct pronunciation	#	phon. respel.	#
Moody	2	m o o d i	0	m uu d i	4	*múdy, můdy*	2
Descartes	2	d e s c a r t e s	0	d e k aa r t	4	*dekárt*	2
Weinfurter	30	v e j n f u r t e r	13	v a j n f u r t r	19	*vajnfurtr*	2
Michelangelo	6	m i x e l a ng g e l o	2	m i k e l a n dzh e l o	4	*mikelandželo*	0

Table 2: Examples of non-standard pronunciation in the manually transcribed data

	phonetically correct	phonetically incorrect
orthographically correct	25	15
orthographically incorrect	6	0

Table 3: Success rate for phonetic and orthographic representation of foreign words based on data from table2

This work has been using language resources stored and distributed by the LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (project LM2015071).

References

[1] Jurik Hájek. Český mystik Karel Makoň. *Dingir*, 2007/4:142–143, 2007.

[2] Andreas Stolcke. Srilm-an extensible language modeling toolkit. In *Seventh international conference on spoken language processing*, 2002.

[3] Jan Hajič. Complex corpus annotation: The prague dependency treebank. *Insight into Slovak and Czech Corpus Linguistics. Veda Bratislava*, 2005:54–73, 2005.

[4] Rada Mihalcea and Timothy Chklovski. Building sense tagged corpora with volunteer contributions over the web. *Recent Advances in Natural Language Processing III: Selected Papers from RANLP 2003*, 260:357, 2004.

[5] Samuel Reese, Gemma Boleda, Montse Cuadros, and German Rigau. Wikicorpus: A word-sense disambiguated multilingual wikipedia corpus. 2010.

[6] Nancy Ide, Christiane Fellbaum, Collin Baker, and Rebecca Passonneau. The manually annotated subcorpus: A community resource for and by the people. In *Proceedings of the ACL 2010 conference short papers*, pages 68–73. Association for Computational Linguistics, 2010.

[7] Jan Nouza, Josef Psutka, and Jan Uhlír. Phonetic alphabet for speech recognition of czech. *Radioengineering*, 6(4):16–20, 1997.

[8] Josef Psutka, Jan Hajic, and William Byrne. The development of asr for slavic languages in the malach project. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, volume 3, pages iii–749. IEEE, 2004.

Computational Intelligence and Data Mining – 6th International Workshop (CIDM 2018)

As a part of the conference ITAT 2018, the 6th international workshop “Computational Intelligence and Data Mining” has been organized. It is aimed at participants with research interests in any of these related areas, especially at PhD students and postdocs. Interested participants were invited to submit a paper in English of up to 8 double-column pages, prepared according to the instructions at the ITAT 2018 web pages.

As this workshop started, 5 years ago, it had only 7 regular submissions. However, the interest in the computational intelligence and data mining workshops has been gradueally incereasing since that time. This year, we have 1 invited talk by internationally renowned expert Filip Šroubek, and 12 regular papers had been submitted to the workshop, among which 11 have been accepted for oral presentations and for inclusion in these proceedings.

A key factor influencing the overall quality of a workshop and of the final versions of the submitted papers is the workshop’s program committee. The 6th international workshop “Computational Intelligence and Data Mining” is grateful to the 27 reviewers from 11 countries who read the submitted papers, and have provided competent, and in most cases very detailed, feedback to their authors. Most of them have a great international reputation witnessed by hundreds of WOS citations.

Workshop Program Committee

Dirk Arnold, University of Dalhousie
Jose Luis Balcazar, Technical university of Catalonia, Barcelona
Petr Berka, University of Economics, Prague
Hans Engler, University of Georgetown
Jan Faigl, Czech Technical University, Prague
Pitoyo Hartono, University of Chukyo
Martin Holeňa, Czech Academy of Sciences, Prague
Tamás Horváth, Eötvös Loránd University, Budapest
Ján Hric, Charles University, Prague
Jan Kalina, Czech Academy of Sciences, Prague
Jiří Kléma, Czech Technical University, Prague
Tomas Krilavičius, Vytautas Magnus University, Kaunas
Věra Kůrková, Czech Academy of Sciences, Prague
Stéphane Lallich, University of Lyon

Philippe Lenca, Telecom Bretagne, Brest
Johannes Lengler, ETH Zürich
Antoni Ligęza, AGH University of Science and Technology, Kraków
Mirko Navara, Czech Technical University, Prague
Engelbert Memphu Nguifo, Blaise Pascal University, Clermont-Ferrand
Ostap Okhrin, Technical University of Dresden
Tomáš Pevný, Czech Technical University, Prague
Petr Pošík, Czech Technical University, Prague
Jan Rauch, University of Economics, Prague
Heike Trautmann, University of Münster
Tingting Zhang, Mid-Sweden University, Sundsvall
Filip Železný, Czech Technical University, Prague

CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Filip Šroubek

Digital image restoration: blur as a motion cue

Filip Šroubek

AVČR

Abstract: We rely on images with ever growing emphasis. Our perception of the world is however limited by imperfect measuring conditions and devices used to acquire images. By image restoration, we understand mathematical procedures removing degradation from images. Two prominent topics of image restoration that has evolved considerably in the last 10 years are blind deconvolution and superresolution. Deconvolution by itself is an ill-posed inverse problem and one of the fundamental topics of image processing. The blind case, when the blur kernel is also unknown, is even more challenging and requires special optimization approaches to converge to the correct solution. Superresolution extends blind deconvolution by recovering lost spatial resolution of images.

In this talk we will cover the recent advances in both topics that pave the way from theory to practice. Various real acquisition scenarios will be discussed together with proposed solutions for both blind deconvolution and superresolution and efficient numerical optimization methods, which allow fast implementation. Finally we will illustrate that combing deblurring with tracking leads to interesting applications in videos.

Filip Šroubek is currently with the Institute of Information Theory and Automation, the Czech Academy of Sciences, and lectures at Charles University. From 2004 to 2006, he was on a postdoctoral position in the Instituto de Optica, CSIC, Madrid, Spain. In 2010 and 2011, he was the Fulbright Visiting Scholar at the University of California, Santa Cruz. His research covers all aspects of image processing, in particular, image restoration (denoising, blind deconvolution, super-resolution) and image fusion (multimodal, multifocus). He is an author of 8 book chapters and over 60 journal and conference papers.

S. Krajči (ed.): ITAT 2018 Proceedings, pp. 44–51
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Jan Motl and Pavel Kordík

Do We Need to Observe Features to Perform Feature Selection?

Jan Motl, Pavel Kordík

Czech Technical University in Prague,
Thákurova 9, 160 00 Praha 6, Czech Republic,
jan.motl@fit.cvut.cz, pavel.kordik@fit.cvut.cz

Abstract: Many feature selection methods were developed in the past, but in the core, they all work the same way — you pass a set of features to the algorithm and get a reduced set of the features. But can we perform a non-trivial feature selection without first observing the features? This is an important question because if we were actually able to predict feature importance before observing the features, we would reduce computation requirements of all stages of machine learning process beginning with feature engineering. In this article, we argue that it is possible to predict feature importance before feature vector observation. The trick is that we use meta-features about the features to perform the feature selection. We evaluate the concept on 15 relational databases. On average, it was enough to generate the top decile of all features to get the same model accuracy as if we generated all features and passed them to the model.

Keywords: meta-learning, feature engineering, feature selection, relational database, propositionalization

1 Introduction

Data in relational databases are in the form of many tables, but common classification algorithms require input data in the form of a single table. Propositionalization solves this discrepancy by converting data from the form of many tables into a single table.

But there are two significant problems with the propositionalization [3]. It produces a lot of features. And many of them are redundant. These two issues result in high computational requirements during both, propositionalization and classification.

Contrary to the common approach (e.g., [10], [7], [8]), we deal with these two issues by performing feature selection *before* the propositionalization and not *after* the propositionalization. The key idea is that we collect meta-data about the attributes in the database (e.g., attribute data type), meta-data about the feature generative functions (e.g., id of the feature function), calculate landmarking features on a small subset of all features and pass their performance to a meta-learner, which predicts the optimal order, in which the remaining features should be calculated.

2 Related Work

The presented work is at the border between feature engineering and feature selection. Hence, we review related work from both these disciplines.

2.1 Meta-learning for Feature Engineering

Meta-learning was originally concerned with algorithm selection[21]. Nevertheless, Nargesian [16] trained a neural network to predict, which feature transformations are going to improve the accuracy of a classifier based on the feature histograms.

We extend the idea of using the data-based meta-features (in Nargesian's case a histogram) for feature engineering with landmarking.

2.2 Meta-learning for Feature Selection

Reif [20] applies meta-learning to accelerate forward selection. The key concept is that the performance of all candidate feature subsets in each forward step is first estimated with a meta-learner. And only the top x percent of the candidates get evaluated on the data to get the true subset performance. Based on the reported results, it is sufficient to evaluate only the top 10% of all candidate subsets on the data to get results comparable to classical forward selection.

The difference between our approach and Reif's approach is that Reif calculates meta-features from the features, while we calculate meta-features directly from the attributes that are used to calculate the features (in Figure 1 we use only the left table, while Reif uses the right table). Consequently, in Reif's case, we have to calculate the features first, to perform feature selection. While in our case, we can perform the feature selection before feature calculation.

3 Method

A high-level schema of our approach is in Figure 2. The whole process is divided into two phases. During the offline phase, meta-features and feature performance are collected on many databases and passed to a meta-learner as training data. During the online phase, the trained meta-learner is used to rank candidate features in the descending order of their estimated utility. Following paragraphs define the *feature utility*.

There are many properties that a feature should posses [12], but we focus on predicting properties measurable directly from the data: relevance to the task, redundancy to other features and runtime of the feature calculation.

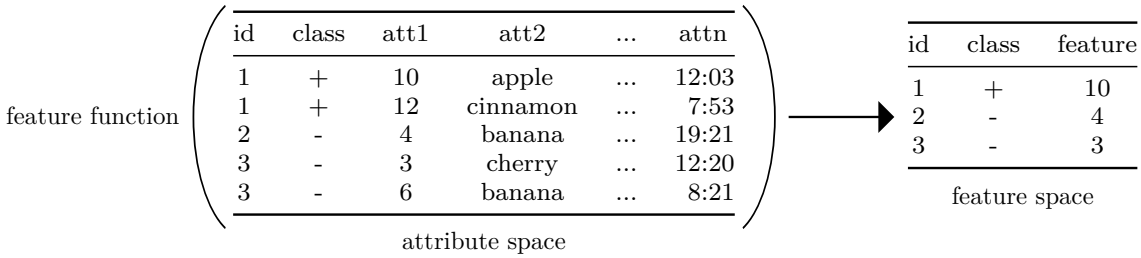


Figure 1: An example of a feature generative function *min* applied on attribute *att1*, which converts the multi-instance problem into a single-instance problem solvable with a common attribute value classifier. In this trivial example, the feature space contains only a single feature vector but it may generally contain thousands of feature vectors.

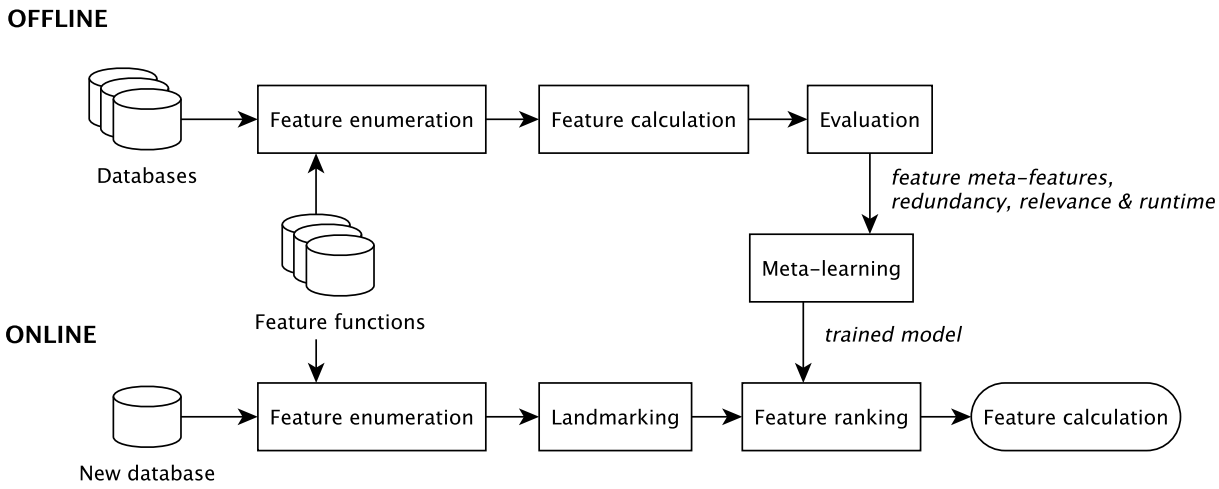


Figure 2: Flowchart of meta-learning on features.

Relevance Without loss of generality, we assume that we want to utilize the calculated features for classification. We use $Chi^2 statistics$ [4, Section A.6.1] between the feature and the label as the measure of the relevance (if the feature is continuous, we first discretize the feature with equal-width binning). But in theory, any other measure can be used.

Runtime The runtime is defined as the time needed to calculate a particular feature vector. If two feature vectors are otherwise identical, we prefer the one that has a smaller runtime.

Redundancy In the analyzed databases (discussed further in Section 4.1), 38% of all calculated features are redundant. We define that nominal feature f_1 is *redundant* to nominal feature f_2 iff a bijection exists between values in f_1 and f_2. A numerical feature f_1 is redundant to numerical feature f_2 if a linear transformation from f_1 to f_2 and back exists.

We use this (weaker) definition of redundancy instead of the *identity* of the features because it corresponds better with the notion of redundancy in many models (e.g., in logistic regression with one shot encoding of categorical features). To speed up the identification of redundant features, we use Chi^2 as a hash function to identify potential redundant features [19, Section 2.1].

3.1 Feature Utility

We calculate features[1] in descending order of the estimated $relevance/runtime$ ratio [1] since we prefer to calculate highly relevant and fast features first. Furthermore, we penalize the feature i proportionally to the estimated probability that the feature is redundant $\hat{p}_i$. Because each dataset has a different proportion of redundant features (see Table 1) and the tested meta-learning models had difficulties to model these differences, we employ median thresholding instead of a fixed threshold:

$$utility_i = (\hat{p}_i > median(\hat{p})\,?\,1-\hat{p}_i\,:\,1)\frac{relevance_i}{runtime_i}, \quad (1)$$

where *redundancy* is a vector of estimated redundancy probabilities for a database.

[1] In the production, we would calculate only the top n features that we would use to build a production classifier. But to demonstrate the meaningfulness of such approach, we calculate all features.

4 Experiment

4.1 Data

We used 15 databases listed in Table 1 from relational repository [15].

4.2 Features

For propositionalization, we used Relaggs [9], which was modified to work with 31 different feature (generative) functions, listed in Figure 2. The detail description of the employed feature functions is at `http://predictorfactory.com`.

4.3 Meta-features

We employ three sources of meta-features: landmarking features, database meta-data and feature function meta-data.

Landmarking features Just like the accuracy of a few classifiers can be used as meta-features for the recommendation of the best classifier on the data (e.g., [18]), we define a subset of feature functions as landmarking feature functions for the recommendation of the best features.

Without loss of generality, we used following set of landmarking features: Direct field (a simple copy of the value), Aggregate (e.g., *min*, *max*,...), WOE (Weight of Evidence), Count (of tuples), Aggregate WOE, Time aggregate since. These feature functions were selected for their low runtime (see Table 11 in the appendix) and good coverage of different data types (numerical/character/temporal) and relationships between the label and the data (1:1/1:n). Note that we do not use multivariate feature functions for landmarking due to the potential combinatorial explosion.

Database meta-data Basic descriptive and statistical meta-features are frequently employed in meta-learning (e.g., [11]) and we do not differ in this respect. A noteworthy difference is that we do not calculate statistics of the attributes but rather reuse statistics maintained by the relational database for query plan optimization [14]. This slight deviation allows us to collect estimates of the statistics in time independent on the count of tuples (records) in the database.

Feature function meta-data Feature function meta-data consists of feature function name (e.g., Aggregate) and feature function parameters (e.g., *min*).

4.4 Measures

Anytime algorithm We formulate feature engineering as anytime algorithm [25], which aims to deliver the best subset of calculated features in any time. The quality of anytime algorithm can be expressed with a performance profile, where we measure quality of the solution at the given time (see example in Figure 3). To assign a single number to the performance profile, we calculate the area between the *archived* curve $a(t)$ and the expected *random* curve $r(t)$ (which we obtain from averaging the curve from many random permutations), divided by the area between the *perfect* curve $p(t)$ and the expected random curve $r(t)$:

$$POP = \frac{\int a(t)dt - \int r(t)dt}{\int p(t)dt - \int r(t)dt}, \tag{2}$$

where t is time. The obtained ratio then represents the "percentage of perfect" solution [2]. In our case, $a(t)$, $r(t)$ and $p(t)$ are the Chi^2 of the feature calculated at time t. The only difference between these functions is then the order, in which the features are calculated. The perfect feature ordering is based on a complete knowledge of relevance, redundancy and runtime of all the features. While archived ordering is based only on the estimates of these feature properties (the only exception are landmarking features, which are calculated in a pseudorandom order).

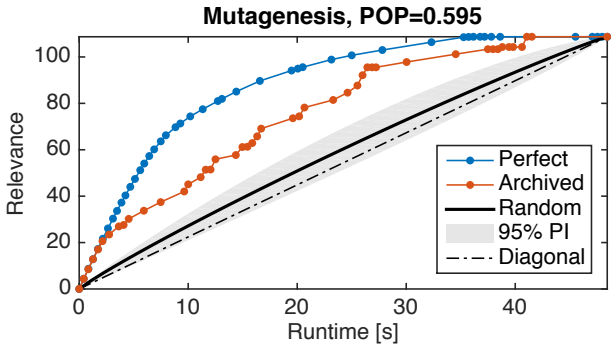


Figure 3: Performance profile. The shaded area represents the 95% prediction interval for a random curve.

Individual models To assess the ability of relevance and runtime prediction models to rank, we use Spearman correlation (ρ). The quality of redundancy estimation (a classification task) is evaluated with area under receiver operating characteristic curve (AUROC).

4.5 Methodology

Algorithms were evaluated with leave-one-out validation, where all but the tested database was used for the training of the models. Obtained accuracies are reported for three different algorithms: *generalized linear model* (GLM), *gradient boosting machine* (GBM) and *deep learning* (DL), all from H2O.

Permutation testing To assess, whether the obtained performance profiles are significantly better than random, we generate 1,000 random orderings of the features to estimate 95% prediction intervals.

Table 1: Used databases. The range of relevant features is estimated with forward & backward selection with a decision tree (the percentage of features when meta-learning feature selection reaches accuracy corresponding to accuracy obtained on all the features).

Database	Domain	Attributes	Features	Redundant [%]	Relevant [%]
Accidents	Government	43	305	39	2–12 (7)
AustralianFootball	Sport	77	794	45	1–6 (1)
BasketballMen	Sport	195	865	41	1–49 (1)
Biodegradability	Medicine	17	71	25	6–66 (4)
Chess	Sport	45	127	16	65–72 (91)
Financial	Financial	55	493	32	1–59 (7)
Hepatitis	Medicine	26	152	42	4–42 (5)
Mondial	Geography	167	1524	45	1–9 (1)
Mutagenesis	Medicine	14	65	40	6–46 (6)
Nations	Geography	118	191	76	2–21 (3)
PremierLeague	Sport	217	667	23	2–27 (7)
PTE	Medicine	76	691	58	1–33 (1)
StudentLoan	Education	15	41	7	15–66 (21)
VisualGenome	Education	20	42	64	10–10 (14)
Walmart	Retail	27	545	19	1–22 (4)
average		74	438	38	8–36 (11)

5 Results

First, we report the accuracy of the individual models. Second, we comment on the meta-feature importance as reported by L1 & L2 regularized GLM. Third, we report the obtained POPs.

5.1 Accuracy

The obtained accuracies are depicted in Table 3. Since the difference between the models is not significant, we use GLM for all following experiments.

5.2 Feature Importance

Relevance The most important meta-features for feature relevance prediction is the average relevance of the landmarking features on individual attributes and the type of the employed feature function (see Table 4).

Redundancy There are two main sources of redundant features [10]: redundancy in the input data and redundancy introduced by the feature functions. The redundancy in the input data is covered by landmarking *landmark_is_redundant* and *data_type*. While the introduced redundancy is explained with *feature_function* and *feature_parameters* (see Table 5).

Runtime The runtime of a feature function calculation is a function of two factors: the type of the feature function and data property. Nevertheless, these two factors are dominated by the landmarking *landmark_runtime* (see Table 6).

5.3 Percentage of Perfect

The quality of anytime learning for all 15 datasets is reported in Table 7 in the penultimate column.

6 Discussion

6.1 What is the contribution of the individual models to POP?

To evaluate the contribution of the individual models to POP, we performed an experiment with a 2-level full factorial design for presence/absence of runtime, relevance and redundancy models (8 combinations in total) on all databases. To deal with the variability across databases (some are easier than others), we treat the database name as a random factor.

Conclusion: The result of the factor analysis is in Table 8. As expected, the intercept is not significantly different from zero, since POP measure should on average be 0 when we randomly rank the features. The biggest contributions to the accuracy are from redundancy and relevance prediction. The interaction between redundancy and relevance has a negative estimate because we do not reward calculation of redundant features even if they are highly relevant. Hence, prediction of the relevance helps only on the subset of unique features from the set of all candidate features.

6.2 What is the effect of meta-learning on model accuracy?

To evaluate the effectivity of the meta-learning, we iteratively train a classification model on increasing percentage

Table 2: Taxonomy of feature functions (data type they work on: c-character, n-numeric, t-temporal). The horizontal axis differentiates between feature functions working on a single attribute and multiple attributes. The vertical axis differentiate between feature functions working on a single tuple and multiple tuples.

	Univariate		**Multivariate**
1:1	Direct field (any) Text length (c) Time day part (t) Time is weekend (t) Time part (t) Time since (t) WOE (c)		Time diff (t+t)
1:n	Aggregate (n) Aggregate distinct (n) Aggregate range (n) Aggregate text length (c) Aggregate WOE (c) Coefficient of variation (n) Count (any) Distinct count (any) Duplicate ratio (any)	Existential count (any) Log product (n) Null ratio (any) Time aggregate (t) Time aggregate since (t) Time aggregate since event (t) Time frequency (t) Time range (t) Time WOE (t)	Aggregate frame (n+t) Correlation (n+t) Intercept (n+t) Slope (n+t) Time aggregate diff (t+t)

Table 3: Leave-one-out accuracy of individual models.

Algorithm	Relevance [ρ]	Runtime [ρ]	Redundancy [AUROC]
DL	**0.558** $\pm$ 0.255	0.302 $\pm$ 0.186	0.798 $\pm$ 0.097
GBT	0.556 $\pm$ 0.252	0.206 $\pm$ 0.259	0.787 $\pm$ 0.106
GLM	0.551 $\pm$ 0.276	**0.369** $\pm$ 0.267	**0.810** $\pm$ 0.102

Table 4: Meta-features for relevance prediction.

Meta-feature	Comment	Weight
landmark_relevance	average on the attribute	9.81
feature_function	e.g., null_ratio is inferior to direct field	5.58
feature_parameters	e.g., aggregate=min is inferior to aggregate=avg	1.90
data_type	e.g., enums are superior to datetimes	0.34
avg_length	extremely long attributes like text are subpar	0.25
is_primary_key	surrogate primary keys make inferior features	0.10

of the top features, as estimated with meta-learning. As the classification model, we use a decision tree because it can model interactions between the features, it is undemanding on data preprocessing and it is reasonably fast. As the evaluation measure, we use misclassification error as all databases have reasonably balanced classes in the label.

An example of the obtained curve is depicted in Figure 4, where we can observe that the decision tree slightly overfits when we use all the features. Nevertheless, forward selection still outperforms meta-learning feature selection,

Table 5: Meta-features for redundancy prediction.

Meta-feature	Comment	Weight
landmark_redundancy	average on the attribute	16.70
feature_parameters	e.g., aggregate=min is inferior to aggregate=avg	13.65
feature_function	e.g., null_ratio is inferior to count	2.61
data_type	e.g., integers are inferior to doubles	0.02

Table 6: Meta-features for runtime prediction.

Meta-feature	Comment	Weight
landmark_runtime	average on the attribute	5.00
feature_function	complicated features take more time	3.34
table_rows	more data means higher runtime	0.10

as it can observe all the features (our approach does not) and it is a wrapper (our approach is a filter [5]).

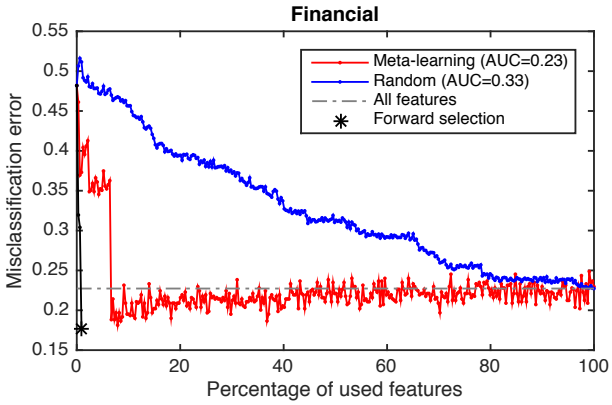


Figure 4: Area under misclassification error. Smaller is better.

Conclusion: The result of the factor analysis is in Table 9. Prediction of relevance significantly reduces misclassification error. Prediction of runtime insignificantly increases the misclassification error, because this evaluation does not reward fast features. Redundancy prediction does not significantly decrease the classification error. Based on our inspection of the results, this is because this evaluation rewards early discovery of a few highly relevant features much higher (since the best possible decision tree may use just a few features) than it penalizes redundancy (a redun-

Table 7: POPs for all databases based on the used individual models. PI column contains the upper 95% prediction interval of POPs for random ordering of the features. The best values are in bold.

redundance	0	1	0	0	1	1	0	1	
relevance	0	0	1	0	1	0	1	1	
runtime	0	0	0	1	0	1	1	1	PI
Accidents	0.11	0.45	0.61	–0.18	**0.68**	0.44	0.60	0.67	0.33
AustralianFootball	0.03	0.33	0.35	–0.09	0.40	0.33	0.35	**0.40**	0.36
BasketballMen	0.08	0.53	–0.52	0.28	0.62	0.56	–0.37	**0.62**	0.11
Biodegradability	–0.18	0.31	–0.44	0.24	**0.34**	–0.21	–0.32	0.32	0.31
Chess	0.05	0.46	0.72	0.31	0.71	0.80	**0.90**	0.87	0.29
Financial	–0.01	0.24	–0.02	–0.01	0.30	0.29	0.02	**0.35**	0.32
Hepatitis	0.07	0.37	–0.01	0.03	0.34	**0.48**	0.04	0.37	0.28
Mondial	–0.00	0.19	0.30	–0.09	0.32	0.27	0.26	**0.32**	0.11
Mutagenesis	0.05	0.14	0.09	0.20	0.24	**0.63**	0.62	0.59	0.22
Nations	0.16	0.59	0.87	0.18	0.75	0.79	**0.88**	0.76	0.34
PremierLeague	–0.07	0.12	0.27	0.08	0.17	0.35	**0.35**	0.34	0.17
PTE	0.01	0.39	0.32	–0.43	0.54	0.31	0.24	**0.53**	0.20
StudentLoan	0.25	0.17	0.62	0.14	0.61	0.53	**0.65**	0.61	0.39
VisualGenome	–0.11	0.44	0.95	–0.03	0.95	0.74	**0.96**	0.94	0.82
Walmart	–0.18	0.20	0.77	–0.06	0.49	0.17	**0.81**	0.52	0.42
average	0.02	0.33	0.32	0.04	0.50	0.43	0.40	**0.55**	0.31
win count	0	0	0	0	2	2	**6**	5	0

Table 8: Contribution of models to POP. Adjusted R^2: 0.564.

	Estimate	Std. Error	Pr(> |t|)	
(Intercept)	0.020	0.0586	0.6877	
relevance	0.292	0.0149	4.0527×10^{-5}	***
redundance	0.318	0.0149	2.9095×10^{-5}	***
runtime	0.053	0.0122	0.0124	*
rel:red	−0.1723	0.0244	0.0021	**

dant feature only pushes all subsequent features one step later).

Table 9: Contribution of models to reduction of the area under misclassification error curve. Adjusted R^2: 0.486.

	Estimate	Std. Error	Pr(> |t|)	
(Intercept)	0.294	0.020	1.3357×10^{-5}	***
relevance	−0.053	0.016	0.0016	**
redundance	−0.018	0.014	0.2402	
runtime	0.010	0.014	0.5167	

6.3 Which meta-features are important?

To analyze the importance of the three categories of the meta-features (*landmarking*, *database*, *feature-function*), we design an experiment, in which we vary the set of the used meta-features.

Table 10: Contribution of meta-feature categories to the reduction of the count of engineered features needed to reach or surpass model accuracy obtained on a complete set of features. Adjusted R^2: 0.308.

	Estimate	Std. Error	Pr(> |t|)	
(Intercept)	43.328	11.569	0.0013	**
database	−0.741	11.007	0.9472	
featureFunction	−3.755	11.007	0.7377	
landmarking	−30.586	11.007	0.0140	*

Conclusion: Based on the results reported in Table 10, only landmarking meta-features help to significantly[2] reduce the count of features that have to be engineered to reach model accuracy obtained on all features. Table 10 also tells us that if all meta-features are used, it is in average sufficient to engineer only the top 8.25% of the features to match or surpass the classification accuracy of the model trained on all features.

6.4 Do we need so many feature functions?

We may wonder whether it is not enough to just engineer the 6 *landmarking features* and do not continue with the engineering of the remaining 25 (e.g., multivariate) features. We compared accuracies of the models trained only on the landmarking features with accuracies obtained on all

[2]The reported p-values do not incorporate correction for repeated evaluation of serially correlated observations

features. Based on Wilcoxon signed-rank test, we have to reject the null hypothesis that the additional features do not improve accuracy (p-value = 0.00048). The median improvement is 1.2 percent point in classification accuracy (average improvement is 2.7 percent point).

Conclusion: The additional features improve the accuracy of the model over the accuracy of the model build only on the landmarking features by a small but significant amount.

6.5 Feature Selection vs. Feature Meta-learning

The described feature meta-learning bears similarity with filter-type feature selection methods like *Correlated Feature Selection* (CFS)[6] and *Minimum Redundancy Maximum Relevance* (mRMR)[17]. Both these methods attempt to quickly select relevant non-redundant features. And so does our method. But in comparison to these methods, we perform feature selection before the feature engineering.

Difficulty It can be argued that feature meta-learning is at least as difficult problem as feature selection since we can always convert feature selection problem to feature meta-learning by throwing away the computed features (and recalculating them on request).

6.6 Limitations

We performed experiments only on relational data and features from propositionalization. Propositionalization is known to produce a lot of duplicate features (38% on average on the tested databases) and many of the features are irrelevant to the task (64% on average on the tested databases based on backward selection). These properties make it possible to obtain substantial gains from feature selection. However, the performed experiments do not tell us how the described approach is going to generalize on non-relational data.

Another limitation of the reported work is that it ignores interactions between the feature vectors in the downstream model. This can reduce the accuracy of the downstream model because a univariate oraculum meta-learner would not recommend calculation of features that are useful only in the combination with other features (a trivial example where this may happen is XOR problem [13]). Possible solutions to this problem are briefly mentioned in the future work Section 7.

6.7 Applications

Feature meta-learning is desirable in domains, where a single universal approach to feature extraction does not exist or is not known ahead. An exemplary domain are relational data, which may contain highly diverse content ranging from structured to unstructured data.

Additionally, feature selection before feature engineering is applicable to complex or large data, where it is not feasible or convenient to calculate and evaluate all possible features due to limited resources.

7 Conclusion

In this article, we evaluated an idea of performing feature selection *before* feature engineering. To guide the search, we exploited *meta-learning*. Nargesian [16] used meta-features calculated from the original data. But we found out that landmarking meta-features work better. When we evaluated the implementation on 15 databases, we concluded that it is on average enough to engineer only *the top decile* of features to get accuracy comparable to accuracy obtained on all features. This finding is similar to Reif's [20] finding, who applied meta-learning to feature selection. However, Reif performs feature selection *after* feature engineering while we perform feature selection *before* feature engineering.

7.1 Future Work

In this exploratory work, we optimize an ersatz measure called POP, which is easy to reason about. One possible extension of this work is to improve individual components of the meta-learning model. For example, we can detect redundancy based on a fuzzy comparison of equal-height histograms estimated with the database engine or quantile sketch. With this change, we would detect duplicates that are identical up to a monotonic transformation, leading to better alignment with models that are invariant to monotonic transformations of the features (e.g., decision trees in theory). Or we could replace the redundancy detection with a precomputed correlation matrix describing similarities between the feature functions [24, p. 148]. Alternatively, we could estimate a transition matrix describing the optimal order in which to apply feature functions (or give up on the given attribute). To improve non-redundancy and relevance together, we could train a fast model (e.g., naive Bayes) on streaming features (e.g., [23] or [22, p. 19]). The possibilities are vast.

Another possible direction is to directly optimize the measure we are interested in (e.g., improvement to model's AUROC over time). This can be done by training a single model (e.g., [16]). And this article provides an extended set of meta-features, on which such model could be trained.

8 Acknowledgement

We would like to thank Adéla Blažková for her help. We furthermore thank the anonymous reviewers, their comments helped to improve this paper. The reported research has been supported by the Grant Agency of the Czech Technical University in Prague (SGS17/210/OHK3/3T/18) and the Czech Science Foundation (GAČR 18-18080S).

References

[1] S. M. Abdulrahman and P. Brazdil. Measures for combining accuracy and time for meta-learning. *CEUR Workshop Proc.*, 1201:49–50, 2014.

[2] T. Brandenburger and A. Furth. Cumulative Gains Model Quality Metric. *J. Appl. Math. Decis. Sci.*, 2009:1–14, 2009.

[3] L. De Raedt. *Inductive Logic Programming*, volume 1446 of *Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg, 1998.

[4] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley Interscience, 2 edition, 2000.

[5] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. An Introduction to Variable and Feature Selection Isabelle. *Mach. Learn.*, 46(1/3):389–422, 2002.

[6] M. A. Hall. *Correlation-based Feature Selection for Machine Learning*. PhD thesis, The University of Waikato, 1999.

[7] J. M. Kanter. Deep Feature Synthesis: Towards Automating Data Science Endeavors. *IEEE DSAA*, 2015.

[8] C. S. Kheau, R. Alfred, and L. H. Keng. Dimensionality Reduction in Data Summarization Approach to Learning Relational Data. *ACIIDS*, 7802:166–175, 2013.

[9] M.-A. Krogel and S. Wrobel. Transformation-Based Learning Using Multirelational Aggregation. In *ILP*, pages 142–155. Springer, London, 2001.

[10] M.-A. Krogel and S. Wrobel. Propositionalization and Redundancy Treatment. In *Databases, Doc. Inf. Fusion*, Hannover, 2002. CEUR.

[11] C. Lemke, M. Budka, and B. Gabrys. Metalearning: a survey of trends and technologies. *Artif. Intell.*, 44(1):117–130, 2015.

[12] A. McNab and D. A. Ladd. Information quality: The importance of context and trade-offs. In *Proc. Annu. Hawaii Int. Conf. Syst. Sci.*, pages 3525–3532, 2014.

[13] M. Minsky and S. A. Papert. *Perceptrons. An Introduction to Computational Geometry*. MIT, jan 1969.

[14] J. Motl and P. Kordík. Foreign Key Constraint Identification in Relational Databases. In *ITAT*, pages 106–111. CEUR, 2017.

[15] J. Motl and O. Schulte. The CTU Prague Relational Learning Repository. *arXiv*, page 7, nov 2015.

[16] F. Nargesian, H. Samulowitz, U. Khurana, E. B. Khalil, and D. Turaga. Learning Feature Engineering for Classification. *IJCAI*, (August):2529–2535, 2017.

[17] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE TPAMI*, 27(8):1226–1238, aug 2005.

[18] B. Pfahringer, H. Bensusan, and C. Giraud-Carrier. Meta-Learning by Landmarking Various Learning Algorithms. In *ICML*, volume 951, pages 743–750, 2000.

[19] A. Popescul and L. H. Ungar. Structural Logistic Regression for Link Analysis. *MRDM*, (August):92–106, 2003.

[20] M. Reif and F. Shafait. Efficient feature size reduction via predictive forward selection. *Pattern Recognit.*, 47(4):1664–1673, 2014.

[21] J. Rice. The Algorithm Selection Problem. *Adv. Comput.*, 15(C):65–118, 1976.

[22] J. Tang, S. Alelyani, and H. Liu. Feature Selection for Classification: A Review. *Data Classif. Algorithms Appl.*, pages 37–64, 2014.

[23] K. Yu, W. Ding, D. Simovici, H. Wang, J. Pei, and X. Wu. Classification with Streaming Features: An Emerging-Pattern Mining Approach. *ACM TKDD*, 9(4):1–31, 2015.

[24] Z. Zhou and H. Liu. *Spectral Feature Selection for Data Mining*. CRC, New York, NY, 2011.

[25] S. Zilberstein. Using Anytime Algorithms in Intelligent Systems. *AI Mag.*, 17(3):73–83, 1996.

Appendix

Reproducibility

The used code is published at:
`https://github.com/janmotl/metalearning`.
The used databases are published at:
`https://relational.fit.cvut.cz`.

Table 11: Expected standardized relevance (bigger is better), runtime (smaller is better) and redundancy (smaller is better) of feature functions (sorted by the feature utility).

Feature function	Relevance	Runtime	Redundancy	Utility
Aggregate frame	−2.11	−0.17	0.49	−2.19
Time aggregate diff	−1.53	0.05	0.54	−1.95
Time diff	−0.92	−0.01	0.48	−1.34
Time day part	−1.33	0.02	0.02	−1.31
Time since	−0.89	−0.04	0.45	−1.22
Null ratio	−0.99	−0.02	0.18	−1.06
Time frequency	−0.17	0.35	−0.10	−0.71
Existential count	−0.67	−0.06	0.06	−0.47
Slope	−0.49	0.03	0.03	−0.47
Time WOE	0.51	0.38	0.07	−0.42
Time is weekend	−1.03	−0.12	−0.21	−0.40
Time part	−0.45	0.00	0.05	−0.40
Text length	−0.76	−0.07	−0.09	−0.37
Intercept	1.42	0.36	0.37	−0.21
Correlation	1.32	0.26	0.37	−0.05
Time aggregate	0.57	0.05	0.21	0.19
Aggregate text length	−0.24	−0.05	−0.15	0.29
Aggregate range	0.34	−0.02	0.15	0.34
Duplicate ratio	0.32	0.17	−0.26	0.39
Aggregate distinct	0.56	0.03	0.11	0.44
Time range	0.06	0.06	−0.26	0.45
Coefficient of variation	0.36	0.01	−0.07	0.62
Time aggregate since event	0.19	0.01	−0.24	0.75
Direct field	0.26	−0.05	−0.09	0.79
Distinct count	0.21	−0.04	−0.16	0.82
Aggregate	0.49	0.04	−0.16	0.82
Log product	0.62	−0.02	0.02	0.87
Time aggregate since	1.25	0.27	−0.24	0.95
Count	0.30	−0.08	−0.18	1.13
WOE	1.27	−0.03	−0.01	1.69
Aggregate WOE	1.04	0.01	−0.39	2.05

S. Krajči (ed.): ITAT 2018 Proceedings, pp. 52–58
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Martin Kopp, Lukáš Bajer, Marek Jílek, and Martin Holeňa

Comparing rule mining approaches for classification with reasoning

Martin Kopp[1,2], Lukáš Bajer[2], Marek Jílek[1], and Martin Holeňa[1,3]

[1] Faculty of Information Technology, Czech Technical University in Prague
Thákurova 9, 160 00 Prague
[2] Cisco Systems, Cognitive Research Team in Prague
[3] Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2, 182 07 Prague

Abstract: Classification serves an important role in domains such as network security or health care. Although these domains require understanding of the classifier's decision, there are only a few classification methods trying to justify or explain their results.

Classification rules and decision trees are generally considered comprehensible. Therefore, this study compares the classification performance and comprehensibility of a random forest classifier with classification rules extracted by Frequent Item Set Mining, Logical Item Set Mining and by the Explainer algorithm, which was previously proposed by the authors.

Keywords: Classification; Comprehensibility; Random Forest; Rule Mining

1 Introduction

Classification is one of the main directions of machine learning research. In the past decades, researchers developed classification models which, when learnt properly, can beat humans in tasks we believed they would never succeed, like handwritten text recognition [**?**, **?**], or even tasks that were specifically designed to be unsolvable for computers, like CAPTCHAs [**?**] and other human interaction proofs [**?**].

Current state-of-the-art classifiers excel in predictive performance, but they lack other quality characteristics for human decision process: deep neural networks, for instance, do not provide explanation of their decisions. Yet, the reasoning justifying decisions is critical in many application domains such as medicine or network security. In medicine, physicians would rather believe a less precise model, if the classification is justified by explanation they can understand. In the network security domain, analyst are often overwhelmed by alarms. But investigation without providing context or explanation, why the alarm was raised, takes a substantial amount of time. Current ransomware affairs [**?**] have shown that time is critical when infection occurs.

This work is focused on comparison of context provided by a random forest classifier and three different rule mining approaches. Random forests were chosen as a representative of the state-of-the-art classifiers which is also able to provide a reasonable justification of its decisions. The precision of classifiers based on sets of rules highly depends on the process of mining rules from data. We tested the quality of rules mined by Frequent Item Set Mining [**?**], Logical Item Set Mining [**?**] and extracted by the Explainer algorithm [**?**]. The comparison focuses on the precision and recall over time and also the quality of presented context. The experiments were done on a dataset from the network security domain.

The rest of the work is organized as follows. The next section covers the related work in the field of comprehensible classification. Section **??** briefly introduces all used rule mining approaches that are experimentally evaluated in Section **??**, followed by the conclusion and future plans.

2 Related work

The pressing issue of comprehensible predictive models has been intensively studied in the field of anomaly detection. The first work considering anomaly explanation was [**?**]. It defined an explanation as "provision of a description or an explanation of why an identified anomalous sample is exceptional". The proposed method first detected all distance-based anomalies in the whole attribute space. Then, it identified the smallest subspace in which the anomaly could be still detected and used that subspace as an explanation.

In the approach presented in [**?**], artificial samples are generated in the vicinity of each sample x. Then a classifier is trained to separate the artificial samples from real samples. If x was an anomaly, then artificial samples should be separated easily, which would result in the classifier having low error and vice-versa.

The method proposed in [**?**] derives the anomaly score from the frequency of histogram bins, from which the method also extracts context and explanation of the anomaly.

Authors of [**?**] used the probabilistic RBF kernels to extract and compare feature impact (positive and negative) for each class in multi-class classification problems.

Most of the recent prior art stops the explanation after identifying the set of features by which the sample under investigation can be differentiated from the rest. The Explainer [**?**] describes the sample by a set of association rules.

A comparison by means of comprehensibility of the well established classifiers such as decision trees, nearest neighbours or Bayesian networks was done in [**?**].

The authors of [**?**] compared multiple rule mining approaches, their general similarities and differences and also their computational efficacy. This paper is focused on the comprehensibility and complexity of a context provided to the domain experts by different rule mining approaches.

The most advanced approach we are aware of is [**?**]. It introduced a framework for measuring not only a comprehensibility (if a prediction is presented in a way that it is easy to understand) but also a justifiability (if a model is in line with the domain knowledge). We would like to use this framework in our future work.

3 Rule mining approaches

Association rules are defined as an implication of the form $\mathcal{A} \Rightarrow \mathcal{C}$, where $\mathcal{A},\mathcal{C} \subseteq \mathbf{I}$ and $\mathbf{I} = \{i_1, i_2, ..., i_d\}$ is a set of binary attributes called items. Association rules are typically mined from a database $\mathbf{X} = \{x_1, x_2, ..., x_n\}$. Each line $x \in \mathbf{X}$ represents a set of items $x \subseteq \mathbf{I}$ and is called a transaction. In this paper, we work with a specific type of association rules, often called classification rules, that are also defined as $\mathcal{A} \Rightarrow \mathcal{C}$. Here, $\mathcal{A} \subseteq \mathbf{I}$ and $\mathcal{C} \in \mathbf{C} = \{c_1, c_2, ..., c_k\}$ is a single item representing a particular class.

The rest of this section surveys rule mining approaches used in this paper.

3.1 Frequent item set mining

The Frequent Item Set Mining (FISM) is a general framework for discovering groups of items (item sets) that are often seen together. The first algorithm mining association rules called GUHA was published by Petr Hájek in [**?**]. In fact, it was a mathematical method for automatic search of hypotheses valid in given data, based on generalized quantifiers of Boolean predicate logic, and the quantifier corresponding to association rules was called *founded almost implication*. The very same approach was rediscovered for data mining purposes as the Apriori algorithm by Agrawal almost thirty years later [**?**].

Since then, a variety of improvements for the original Apriori algorithm was proposed [**?**, **?**], and also several new algorithms for frequent items mining were invented, e.g., FP-Growth [**?**], LCM [**?**] or Eclat [**?**]. This paper uses the FP-Growth algorithm as the most time and memory efficient representative of the FISM algorithms.

FP-Growth algorithm starts by building a specific prefix tree called **frequent pattern tree (FP-tree)**. First, the frequencies of all items $i \in \mathbf{I}$ are calculated. Then, all items i with frequency lower than the user specified threshold $\theta_{minFreq}$ are filtered out from all transactions $x \in \mathbf{X}$. Items remaining in the filtered transactions are sorted in descending order according to their frequency. The prefix tree is built by inserting the filtered and sorted transactions.

Once the FP tree is built, it is recursively traversed in a bottom-up manner, mining frequent item sets laying on the path from leaf to root. Based on the FP-tree construction process, each transaction is mapped to a path in the FP-tree. The FP-tree structure also guarantees that all frequent item sets present in the database $\mathbf{X}$ can be found on the path from some leaf to the root. Moreover, one path in the FP-tree may represent frequent item sets in multiple transactions.[1] This property also ensures the memory efficiency of FP-Growth.

3.2 Logical item set mining

The Logical Item Set Mining (LISM) [**?**] is an alternative approach to mining association rules from data. The key difference from FISM is that LISM captures logical relations not only between frequent items, but it also extracts strong relations between rarely occurring items. By leveraging indirect relationships between items, it can also discover relations between item sets that are not present in a dataset. The algorithm has counting, consistency, denoising and discovery phases.

During the **counting phase**, co-occurrence counts, marginal counts and total counts are calculated.

Co-occurrence count $\psi(i_a, i_b)$ for every pair of items $(i_a, i_b) \in \mathbf{I} \times \mathbf{I}$, where $i_a \neq i_b$, is defined as the number of transactions in which both items co-occurred:

$$\psi(i_a, i_b) = \sum_{j=1}^{n} \delta(i_a \in x_j)\delta(i_b \in x_j), \tag{1}$$

where $\delta(condition)$ is an indicator function which is 1 if the *condition* is true and 0 otherwise. The results are stored in the symmetrical matrix $\Phi = [\phi(i_a, i_b)]$, which is usually very sparse.

Marginal count $\psi(i_a)$ is defined as the number of pairs in which the item $i_a \in \mathbf{I}$ occurred with some other item:

$$\psi(i_a) = \sum_{i_b \in \mathbf{I}, i_a \neq i_b} \psi(i_a, i_b). \tag{2}$$

Total count ψ_0 is defined as the total number of pairs in which some item co-occurred with some other item:

$$\psi_0 = \frac{1}{2}\sum_{i_a \in \mathbf{I}} \psi(i_a) = \frac{1}{2}\sum_{i_a \in \mathbf{I}}\sum_{i_b \in \mathbf{I}} \psi(i_a, i_b) \tag{3}$$

These three results are then used as estimates of the co-occurrence and marginal probabilities

$$P(i_a, i_b) = \frac{\psi(i_a, i_b)}{\psi_0}, \quad P(i_a) = \frac{\psi(i_a)}{\psi_0}. \tag{4}$$

The **consistency phase** reduces the effect of noise and amplifies the importance of rare items that are consistently

[1] Items in item sets are ordered in the descending order, frequent items are arranged closer to the top of the FP-tree and more likely to be shared.

seen together. A variety of distance measures can be employed, e.g., cosine, Jaccard or point-wise mutual information. The cosine similarity defined as

$$\phi(i_a, i_b) = \frac{P(i_a, i_b)}{\sqrt{P(i_a)P(i_b)}} \in [0,1] \tag{5}$$

was used in our experiments.

The iterative **denoising phase** uses the co-occurrence consistencies obtained in the previous iteration to remove noisy co-occurrence counts in Φ. Then, marginal and total counts are recomputed solely from the updated Φ, there is no need to touch data again. As a last step of every iteration a consistency counts are updated.

In the **discovery phase**, a graph is created from the denoised co-occurrence consistency matrix Φ. Given Φ, a logical item set is defined as a set of items where each item has a high co-occurrence consistency with all other items in the set. Such sets are found by application of an algorithm for finding maximal cliques, e.g., [**?**], on the co-occurrence consistency graph.

3.3 Explainer

The Explainer [**?**] is a tree based algorithm designed to explain why a sample $x^a \in \mathbf{X}$ is an anomaly with respect to the rest of the data in $\mathbf{X}$. The output can be a set of the most important features or a set of association rules. Properties of extracted rules were studied in [**?**].

To explain an anomaly x^a, the Explainer trains a set of specific decision trees to separate x^a from rest of the data in $\mathbf{X}$. Rules are extracted from each of those trees and assembled into a set of association rules. The key steps are summarized in Algorithm **??**.

Algorithm 1 Summary of the Explainer algorithm for a single anomaly x^a.

Input:
data – input dataset; x^a – anomalous sample; *size* – training set size; n_T – the number of trees to be trained.
Output:
rules – rules explaining x^a

```
Forest ← ∅
for i ← 1 ... n_T do
    T ← createTrainingSet(data, size, x^a)
    t ← trainTree(T)
    Forest ← Forest ∪ t
end for
rules ← extractRules(Forest)
```

During the **training set selection**, a dataset $\mathbf{X} = \{x_1, x_2, \ldots, x_n\}$ is split into two disjoint sets $\mathbf{X}^a$, containing anomalous samples, and $\mathbf{X}^n$, containing normal samples. Then, a training set $\mathbf{T}$ contains the anomaly x^a as one class and a subset of $\mathbf{X}^n$ as the other. The simplest strategy is to select k samples randomly from $\mathbf{X}^n$ with uniform probability. This approach is computationally effective and was proven to work well when compared to more sophisticated approaches [**?**].

Training a tree is very similar to standard random forests [**?**]. A candidate node is found and the optimal splitting function is applied on that node. This greedy procedure repeats until the specified stopping criteria are met.

The node that contain x^a is always the one being split in the Explainer algorithm. The standard procedure to find the splitting function h is maximizing the information gain over the space of all possible splitting functions $\mathcal{H}$. But as there is only a single point x^a in the anomaly class, the information gain is equivalent to minimizing the size of the node containing x^a:

$$\arg\min_{h \in \mathcal{H}} |\mathbf{S}^a(h)|, \tag{6}$$

where $\mathbf{S}^a \subset \mathbf{T}$ denotes the subset of the training set containing the anomaly x^a after the split. The training is stopped when all leaves are pure (contain samples from a single class).

Once a tree is trained, it is used for **rule extraction**. Let $h_{f_1,\theta_1}, \ldots, h_{f_d,\theta_d}$ be the set of splitting functions, with features $f_1, \ldots, f_d$ and threshold $\theta_1, \ldots, \theta_d$, used in inner nodes on a path from the root to the leaf with x^a. Then x^a is explained as a conjunction of atomic conditions:

$$(x_{f_1} > \theta_1) \wedge (x_{f_2} > \theta_2) \wedge \ldots \wedge (x_{f_d} > \theta_d), \tag{7}$$

which is the output of the algorithm. This conjunction can be read as "the sample is anomalous because it is greater than threshold θ_1 in feature f_1 and greater than θ_2 in feature f_2 and ... than majority of samples". Each tree provides one such conjunction, that are then aggregated.

4 Experiments

This section experimentally compares the classification performance of the three described rule mining approaches with the random forest classifier. The section starts with a description of the dataset used in our research and with the setting of the performed experiments. The comparison is then based on the precision and recall measures, and the final part concludes with a thorough discussion of differences of explanations provided by each approach.

4.1 Dataset description

The dataset in this research consists of 8 consecutive weeks of network traffic collected by the Cognitive Threat Analytics (CTA) intrusion detection system. It contains about 9 million samples, where each sample is a collection of all network events observed on a particular network host within the 24-hours window. In the words used in the market basket analysis, each sample x is a transaction containing a subset of all events/items $x \subseteq \mathbf{I}$.

CTA distinguishes about 300 events falling into four categories:

- **signature based (SB)** – produced by matching behavioural signatures handmade by a domain expert.
- **classifier based (CB)** – created by supervised classifiers trained on historical data.
- **anomaly based (AB)** – created by the anomaly detection engine which consists of 70 individual detectors (statistical, volumetric, proximity, targeted, domain specific).
- **contextual events (CE)** – describe various network behaviours to provide an additional context, e.g., file download, direct access on raw IP, software update.

These events together serve as high level behavioural indicators that can be used to create a behavioural profile of a malware family. This behavioural profile can be used to identify malware infections in their early stages and stop them before they do any harm.

The database was labelled by the CTA engine in a way that transactions of a network hosts infected by either banking trojan, click fraud, information stealer or malware distribution were marked as positives/malicious (4801 and 6463 transactions in training and testing sets respectively) and the other transactions were labelled negative/benign (3.75 mil. and 5.23 mil. transactions respectively).

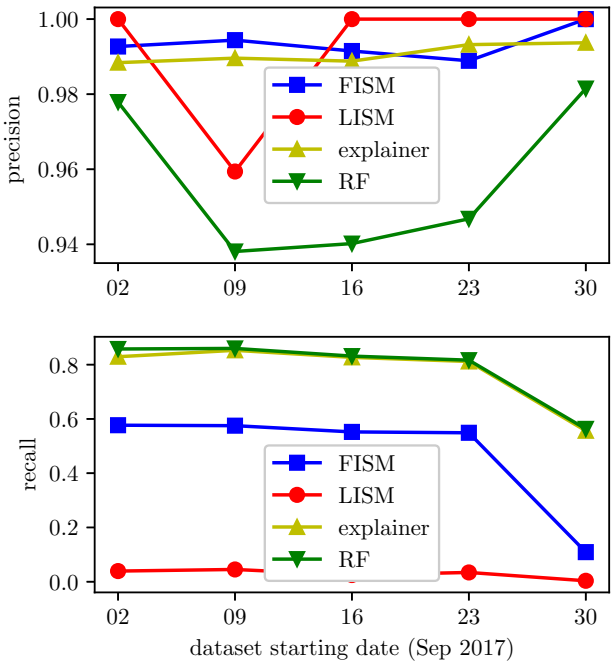


Figure 1: Precision and recall of the four considered classifiers measured in five consecutive weeks in September 2017.

4.2 Experimental setup

All following experiments used 3 weeks of data, approximately 3.75 million of transaction, for training/rule mining and 5 weeks, 5.23 million of transactions, for testing.

Parameters of the random forest were set as follows: number of trees = 19; maximal depth of a tree = 25; number of features per split = 100.

The Explainer was set to train 10 trees per positive sample while selecting a random training set of size 1000 samples.

The FP-growth algorithm was set to produce only rules with support higher than 3 transactions.

The parameters of LISM was: similarity measure = cosine (**??**); $\theta_{cooc} = 1 \cdot 10^{-7}$; $\theta_{cons} = 2$. Cliques discovered using this setting were split into 3–5 items long rules. Details about the optimization of parameters can be found in [**?**].

As a last step, all rules were filtered to have precision on the training set at least 80%. The filtered rules were then used in the following experiments.

4.3 Classification efficacy

The described rule filtering, based on the minimal precision 80% on the training set, resulted in very few false positive predictions: all classifiers reached more than 90%-precision on all the testing sets, as depicted in the upper graph in Figure **??**. While the rules mined by the Explainer and FISM algorithm provide stable results with precision reaching roughly 99% during all five testing weeks, the rules generated by the LISM and random forest exhibit greater variance in time.

The ability to discover the malicious content in the testing data, as measured by the classifiers' recall, diversifies considerably more. Both Explainer's and random forest's sets of rules were able to identify more than 80% of the malicious samples in the testing sets (except the last week). The graph in Figure **??**, on the other hand, clearly shows that the highest precision rate of the LISM-generated rules is accompanied with the lowest degree of recall.

The LISM is able to generate complex and very descriptive rules that, however, can be hardly located in the data. That results in only 34 generated rules reaching the training precision threshold 80%, which is probably the cause behind the lowest recall. The 100%-precision in 4 out of 5 testing weeks is surprisingly good result, though.

The performance of the FISM and Explainer differs mainly in their recall, where the better results of the Explainer is probably caused by the algorithm's focus to the shortest and strongest rules; only 14 rules reached the 80% precision threshold. The FISM, on the other hand, is able to generate longer rules, which are harder to match.

Random forest classifier with its up to 25-levels deep decision trees is able to identify the highest number of the testing malicious transactions. On the contrary, the high complexity of the trees, at the same time, causes the lowest observed precision out of four compared classifiers.

4.4 Context comparison

This section presents the comparison of context provided by the random forest classifier, Explainer, Logical Item Set Mining and Frequent Item Set Mining. Examples of mined rules are presented and discussed in detail.

Random Forest

In general, random forests can provide two types of context; feature importance and classification rules.

The feature importance provides less information than rules. It represents aggregated global knowledge for all classes in the training set. It can be extracted per class if a random forest is trained for each class separately. Our database represents a dichotomy problem with both classes being in fact mixtures of multiple sub-classes that we are unable to separate.

Classification rules provide more specific/local information. Unfortunately, trees in a random forest are typically trained deep to be as precise as possible. In other words, the path from a root to a leaf will be long and therefore, the extracted rule will be also long. The other drawback of training random forests directly on transactions is so called *negative evidence*. In the network security domain, the negative evidence is when a network host is being marked as infected because of some behaviour it doesn't have, e.g. "a host is infected, because it didn't downloaded an image".

The real example of a rule with multiple negative evidences extracted directly from the random forest used in the experiments is: "If you are not infected by a click fraud and you are not infected by an information stealer and you are not infected by the Sality trojan and you are not infected by the malware called Gamarue and you don't have encrypted connection then you are infected." As you can see from this simple yet real example, rules extracted from a random forest may be more puzzling than explaining.

Explainer

The Explainer can be easily set to provide rules with only *positive evidences*. The trouble is, that it was designed to extract the smallest set of the shortest possible rules. The extracted rules contained the real causes of incidents but not any additional context which would simplify the investigation. From the rules created by the Explainer, 14 had a precision higher than 80%. They typically contained only one event produced by a supervised classifier. The second event was presents in only 3 cases and there was no rule longer than that. Selected examples of longer rules follow:

1. AB:ShadowUser CB:ClickFraud
2. CB:SuspAdvertising CB:MaliciousAdvertising

The first example contains an anomaly based event AB:ShadowUser and a classifier based event CB:ClickFraud. AB:ShadowUser identifies network hosts that are visiting a high number of network domains which nobody else visit. The CB:ClickFraud event is created by the random forest classifier trained to discover malware from the click fraud family. The click fraud malware family is known for visiting a lot of weird pages without user's knowledge and is often used for clicking on web advertisements to generate money.

The second rule contains two classifier based events indicating a host visited a lot of suspicious advertisement pages(CB:SuspAdvertising), some of them probably malicious (CB:MaliciousAdvertising). Here, the malware started by showing additional unwanted advertisements, banners and pop-ups and ended by ex-filtrating sensitive data.

LISM

Rules extracted by the Logical Item Set Mining create a very logical and justifiable connections between items. All items in a rule are always very strong indicators of a particular threat. Unfortunately, they would appear all at once very rarely, but if they do, it is for sure a serious infection. LISM created 34 rules with precision over 80%. The length of the rules is ranging from 3 to 5. Examples follow:

1. SB:Blocked CB:MalBinary SB:Sality
2. CB:ClickFraud CB:Malwartising CB:MalwareDistr

The first example shows a download of malicious binary (CB:MalBinary) from a black listed domain (SB:Blocked). Furthermore, it has a well known signature of a malware called Sality (SB:Sality). Each of these events is strong enough evidence to trigger an immediate re-mediation alert on its own.

The second rule shows a nice example of a malicious escalation. At first, a host was infected by a click fraud malware (CB:ClickFraud), which was visiting a shady advertising sites (CB:MalAdvertising). Then, the host started to download and distribute additional malicious modules (CB:MalwareDistr).

FISM

The Frequent Item Set Mining provided the best explanation/context from all compared method. Rules contain all important indicators while providing a reasonable additional context. FISM created 454 rules, with length ranging from 3 to 5, that satisfied 80% threshold on precision, examples follow:

1. CE:jQuery AB:SuspDomain CB:ClickFraud
2. AB:PathCount AB:ShadowUser CB:ClickFraud CE:WPManagment

The first example shows a host that downloaded a modified javascript library jQuery(CE:jQuery) from a suspicious network domain (AB:SuspDomain). This modified library was the source of infection, which was later detected by the random forest classifier trained for detection of a click fraud malware family (CB:ClickFraud).

The second rule shows a sophisticated example of click fraud capabilities (CB:ClickFraud). In that case, the malware had a password cracker module installed and was used to crack into administration sections of web pages created using Wordpress(CE:WPManagment). Again, we can see a host visiting large amount of low probability network domains (AB:ShadowUser), WordPress blogs in that case, where on most of these pages was visited the outlying number of paths (AB:PathCount), specifically only one.[2]

5 Conclusion

This paper compared the classification performance and comprehensibility of a random forest classifier with classification rules extracted by the Frequent Item Set Mining, Logical Item Set Mining and by the Explainer algorithm, which was previously proposed by the authors. All the algorithms showed surprisingly similar precision with rules extracted by LISM performing slightly better, with exception on one week. From the recall point of view the best performing algorithm was the random forest followed by rules extracted by the Explainer.

The comparison of provided explanations revealed that rules extracted from random forests trained on transaction data can be more puzzling than comprehensible. The relationships between items mined by LISM were not only comprehensible but also justifiable (in line with the domain knowledge). Unfortunately, they are very rarely seen within the real data. Rules extracted by the Explainer tend to reveal the root cause of an incident but provide only a little to non additional context. Rules mined by FISM appeared as an optimal mix of root cause events together with a reasonable amount of additional context.

As the future work, we would like to extend our research into the other application domains and also to implement the work of Martens et al. [**?**] and compare their numerical representations of comprehensibility and justifiability with our domain knowledge.

Acknowledgement

The research reported in this paper has been supported by the Czech Science Foundation (GAČR) grant 17-01251 and by Czech Technical University student grant SGS17/209/OHK3/3T/18.

References

[1] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Record*, volume 22, pages 207–216. ACM, 1993.

[2] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, volume 1215, pages 487–499, 1994.

[3] Fabrizio Angiulli, Fabio Fassetti, and Luigi Palopoli. Detecting outlying properties of exceptional objects. *ACM Transactions on Database Systems (TODS)*, 34(1):7, 2009.

[4] Zachary K Baker and Viktor K Prasanna. Efficient hardware data mining with the apriori algorithm on fpgas. In *Field-Programmable Custom Computing Machines, 2005. FCCM 2005. 13th Annual IEEE Symposium on*, pages 3–12. IEEE, 2005.

[5] Christian Borgelt. Efficient implementations of apriori and eclat. In *FIMI03: Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations*, 2003.

[6] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[7] Randy Carraghan and Panos M Pardalos. An exact algorithm for the maximum clique problem. *Operations Research Letters*, 9:375–382, 1990.

[8] Kumar Chellapilla, Kevin Larson, Patrice Simard, and Mary Czerwinski. Designing human friendly human interaction proofs (hips). In *Proceedings of the SIGCHI conference on HumanFactors in Computing Systems*, pages 711–720. ACM, 2005.

[9] Salvador Espana-Boquera, Maria Jose Castro-Bleda, Jorge Gorbe-Moya, and Francisco Zamora-Martinez. Improving offline handwritten text recognition with hybrid hmm/ann models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(4):767–779, 2011.

[10] Alex Alves Freitas. Comprehensible classification models: a position paper. *SIGKDD Explorations*, 15:1–10, 2013.

[11] Petr Hájek, Ivan Havel, and Metoděj Chytil. The GUHA method of automatic hypotheses determination. *Computing*, 1(4):293–308, 1966.

[12] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *ACM SIGMOND Record*, volume 29, pages 1–12. ACM, 2000.

[13] Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh. Algorithms for association rule mining—a general survey and comparison. *ACM SIGKDD Explorations newsletter*, 2(1):58–64, 2000.

[14] Marek Jilek. Machine learning based context extraction for network security incidents. Master's thesis, Faculty of Information Technology, Czech Technical University in Prague, 2018.

[15] Edwin Knorr and Raymond T Ng. Algorithms for mining distancebased outliers in large datasets. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 1998.

[16] Martin Kopp and Martin Holeňa. Evaluation of association rules extracted during anomaly explanation. In *ITAT 2015: Information Technologies - Applications and Theory*, 2015.

[17] Martin Kopp, Matěj Nikl, and Martin Holeňa. Breaking captchas with convolutional neural networks. In *ITAT 2017: Information Technologies - Applications and Theory*, 2017.

[18] Shailesh Kumar, V Chandrashekar, and CV Jawahar. Logical itemset mining. In *Data Mining Workshops (ICDMW),*

[2] When a webpage is opened via a browser it downloads a lot of different directories and files such as cascade style sheets, html file, image folder, javascript libraries, etc.

2012 IEEE 12th International Conference on, pages 603–610. IEEE, 2012.

[19] David Martens, Jan Vanthienen, Wouter Verbeke, and Bart Baesens. Performance of classification models from a user perspective. *Decision Support Systems*, 51(4):782–793, 2011.

[20] Barbora Micenková, Raymond T Ng, Xuan-Hong Dang, and Ira Assent. Explaining outliers by subspace separability. In *IEEE 13th International Conference on Data Mining (ICDM 2013)*, 2013.

[21] Lukáš Neumann and Jiří Matas. Real-time scene text localization and recognition. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3538–3545. IEEE, 2012.

[22] Tomáš Pevný and Martin Kopp. Explaining anomalies with sapling random forests. In *Information Technologies - Applications and Theory Workshops, Posters, and Tutorials (ITAT 2014)*, 2014.

[23] Marko Robnik-Šikonja, Igor Kononenko, and Erik Štrumbelj. Quality of classification explanations with prbf. *Neurocomputing*, 96:37–46, 2012.

[24] Margaret Rouse. Ransomware, 2018. https://searchsecurity.techtarget.com/definition/ransomware [Cited 15 May 2018].

[25] Takeaki Uno, Masashi Kiyomi, and Hiroki Arimura. Lcm ver. 2: Efficient mining algorithms for frequent/closed/-maximal itemsets. In *FIMI*, volume 126, 2004.

[26] Mohammed Javeed Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 12(3):372–390, 2000.

S. Krajči (ed.): ITAT 2018 Proceedings, pp. 59–63
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Jan Motl and Pavel Kordík

Violation of Independence of Irrelevant Alternatives in Friedman's test

Jan Motl, Pavel Kordík

Czech Technical University in Prague,
Thákurova 9, 160 00 Praha 6, Czech Republic,
jan.motl@fit.cvut.cz, pavel.kordik@fit.cvut.cz

Abstract: One of the most common methods for classifier comparison is Friedman's test. However, Friedman's test has a known flaw — ranking of classifiers A and B does not depend only on the properties of classifiers A and B, but also on the properties of all other evaluated classifiers. We illustrate the issue on a question: "What is better, bagging or boosting?". With Friedman's test, the answer depends on the presence/absence of irrelevant classifiers in the experiment. Based on the application of Friedman's test on an experiment with 179 classifiers and 121 datasets we conclude that it is very easy to game the ranking of two insignificantly different classifiers. But once the difference becomes significant, it is unlikely that by removing irrelevant classifiers we obtain a significantly different classifiers but with reversed conclusion.

1 Introduction

Friedman's test is the recommended way how to compare algorithms in machine learning (ML) [9]. It is a nonparametric test, which calculates scores not on the raw performance measures (e.g. classification accuracy or correlation coefficient) but on the ranks calculated from the raw measures. Nonparametric tests are favored over parametric tests in ML, because the standard formulation of the central limit theorem (CLT) does not apply on bounded measures [8], many measures used in ML are bounded, and commonly used parametric tests rely on CLT. Nevertheless, even nonparametric tests, which are based on ranking, have flaws as demonstrated by Arrow's impossibility theorem [1]. In this article, we discuss one such flaw of Friedman's test: violation of *Independence of Irrelevant Alternatives* (IIA).

2 Problem Description

Friedman's test Friedman's test, if applied on algorithm ranking, is defined as:

> Given data $\{x_{ij}\}_{n\times k}$, that is, a matrix with n rows (the *datasets*), k columns (the *algorithms*) and a single performance observation at the intersection of each dataset and algorithm, calculate the ranks *within* each dataset. If there are tied values, assign to each tied value the average of the ranks that would have been assigned without ties. Replace the data with a new matrix $\{r_{ij}\}_{n\times k}$ where the entry r_{ij} is the rank of x_{ij} within dataset i. Calculate then rank sums of algorithms as: $r_j = \sum_{i=1}^{n} r_{ij}$.

We can rank the algorithms based on *rank sums* r_j [10]. Friedman's test then continues with the evaluation of the null hypothesis that there are no differences between the classifiers. Since this article is not about hypothesis testing but rather about algorithm ranking based on r_j, we reference a keen reader to read [9] for a detailed description of Friedman's test hypothesis testing.

IIA Independence of Irrelevant Alternatives [14] condition is defined as:

> If algorithm A is preferred to algorithm B out of the choice set $\{A,B\}$, introducing a third option X, expanding the choice set to $\{A,B,X\}$, must not make B preferable to A.

In other words, preferences for algorithm A or algorithm B, as determined by rank sums r_j, should not be changed by the inclusion of algorithm X, i.e., X is irrelevant to the choice between A and B.

Illustration Boosting tends to outperform base algorithms (e.g. decision trees) by a large margin. But sometimes, boosting fails [2, Chapter 8.2] while bagging reliably outperforms base algorithms on all datasets, even if only by a small margin [2, Chapter 7.1]. This is illustrated in the left part of Figure 1. If we compare boosting only to bagging, then by the rank sums r_j, boosting wins, because boosting is better than bagging on the majority of datasets. However, if we add irrelevant algorithms that are always worse than bagging, the conclusion may change. Bagging will be always the first or the second. But boosting will be either the first or (in the provided illustration) the last. And a few extreme values in the rank sum r_j can result into the change of the conclusion.

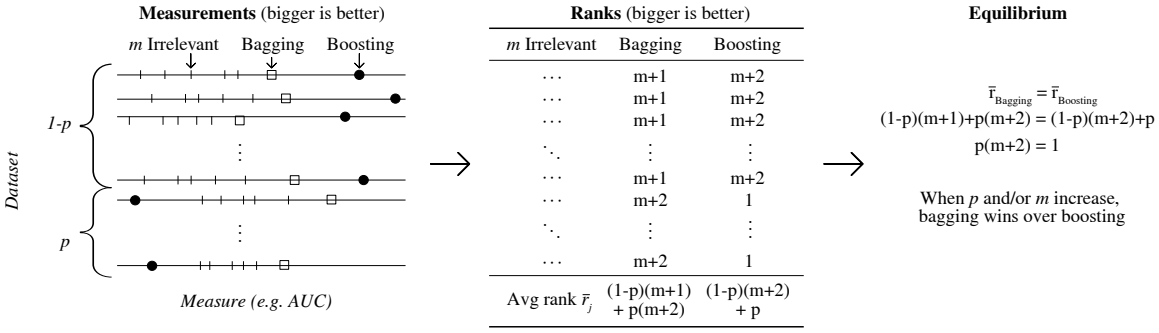


Figure 1: What is better, bagging or boosting? Boosting is better than bagging by a large margin on majority of datasets (e.g. in 95%). But with a constant probability p, boosting fails critically. Interestingly, the decision which of the algorithms is better does not depend only on p but also on the count of irrelevant algorithms (m).

3 Impact

3.1 Bias of Authors

Hypothesis Authors of highly cited papers, knowingly or not, frame their algorithms in the best possible light. Based on the equilibrium equation in Figure 1, we would expect that proponents of boosting algorithms use *fewer algorithms* than proponents of bagging in the design of experiments (DOE).

Evaluation Breiman, the author of bagging [6], compared bagging against 22 other algorithms while Freund and Schapire, authors of AdaBoost [11], compared their boosting algorithm against only 2 other algorithms. Our expectations were fulfilled.

Threats to validity due to omitted variables Since both articles were published in the same year, the availability of algorithms for comparison should be comparable and cannot be used to explain the observed differences in the DOE.

Conclusion Since this is just a single observation, which can be just by chance, following section analyses the impact of DOE numerically.

3.2 Effect on Large Studies

A nice comparison of 179 classifiers on 121 datasets is provided by Fernández-Delgado et al. [10]. The authors follow Demšar's [9] recommendation to use Friedman's test to rank the algorithms.

If we directly compare just two classifiers, *AdaBoostM1_J48_weka* and *Bagging_J48_weka*, and calculate rank sums r_j, then we get that *boosting beats bagging* 64 to 47. But if we calculate rank sums over all algorithms, then we get that *bagging beats boosting* 13136 to 12898. A completely reversed conclusion!

How frequently does the ordering flip? If we perform above analysis over all unique pairs of classifiers ($\frac{1}{2}179(179-1) = 15753$), then we get that the ordering flips in 5% of cases (831/15753).

Are the changes in the ranks significant? We repeated the experiment once again, but considered only classifier pairs that are based on Friedman's test:

1. *pairwise* significantly different
2. and significantly different in the presence of *all the remaining classifiers*.

If we do not apply multiple testing correction, the classifiers flip the order in mere 0.05% cases (8/15753) at a shared significance level $\alpha = 0.05$. Once we add multiple testing correction, *the count of significant flips drops to zero*. In our case, the exact type of multiple testing correction does not make a difference. Bonferroni, Nemeneyi, Finner & Li and Bergmann & Hommel [17] corrections all give the same result as the lowest p-value before the correction in that 8 cases is 0.023, which is already extremely close to the significance level of 0.05.

4 Related Work and Discussion

We are not the first one to notice that Friedman's test does not fulfill the IIA property. The oldest mention of the issue in the literature, that we were able to track down, dates back to 1966 [4]. Following paragraphs discuss possible solutions to the issue.

4.1 Pairwise Ranking

Benevoli et al. [4] recommend replacing Friedman's test with pairwise *Wilcoxon signed-rank test* followed with *multiple testing correction*. The application of a pairwise-test solves the issue with IIA if we want to show how well "a new algorithm A fares in comparison to a set of old algorithms $\mathbb{B}$" because each pairwise comparison between A and some other algorithm $B \in \mathbb{B}$ is by definition independent on $C \in \mathbb{B}, C \neq B$. However, if we aim to "rank the current algorithms together", pairwise tests may not deliver a *total ordering* while a method comparing all algorithms at once can, as illustrated in Figure 2.

	A	B	C
α	1	2	3
β	1	2	3
γ	3	1	2
δ	3	1	2
ε	2	3	1
Rank sum	10	9	11

Figure 2: A minimal example of 3 algorithms and 5 datasets where rank sums deliver total ordering while pairwise methods end up with a cycle: $A \prec B, B \prec C, C \prec A$.

One important implementation detail, which is not discussed by Benevoli et al., is that Wilcoxon signed-rank test does not by default take into account ties (while Friedmans's test does). Ties may appear in an experimental study as a result of rounding or as a consequence of using a dataset with a small sample size. And if the percentage of ties is high, the negligence of the ties can result in misleading results, as discussed by Pratt [13]. In R, we can use `wilcoxsign_test` function from `coin` package, which implements Pratt's tie treatment.

4.2 Vote Theory

During the French revolution in the 18th century, the need to come with a fair vote method arose. Two competitors, Condorcet and Borda[1], came with two different methods. And they did not manage to agree, which of the methods is better. This disagreement spurred an interest into vote theory. One of the possible alternatives to Friedman test that vote theory offers is *Ranked pairs* method [16]:

1. Get the count of wins for each pair of algorithms.
2. Sort the pairs by the difference of the win counts in the pair.
3. "Lock in" the pairs beginning with the strongest difference of the win counts.

While Ranked pairs method also fails IIA criterium, it at least fulfills a weaker criterium called *Local Independence from Irrelevant Alternatives* (LIIA). LIIA requires that the following conditions hold:

> If the best algorithm is removed, the order of the remaining algorithms must not change. If the worst algorithm is removed, the order of the remaining algorithms must not change.

An example where Friedman's test fails LIIA criterium is given in Figure 3 [14].

	A	B	C
α	1	3	2
β	1	3	2
γ	2	1	3
δ	2	1	3
ε	2	1	3
Rank sum	8	9	13

$\longrightarrow$

	A	B
α	1	2
β	1	2
γ	2	1
δ	2	1
ε	2	1
Rank sum	8	7

Figure 3: A minimal example of 3 algorithms and 5 datasets demonstrating that Friedman's test does not fulfill LIIA criterium — when we remove the best algorithm C, algorithm A becomes better than algorithm B.

Friedman's test also violates *Independence of Clones criterion* [15]:

> The winner must not change due to the addition of a non-winning candidate who is similar to a candidate already present.

An example, when can this can become a problem is given by [4]:

> Assume that a researcher presents a new algorithm A_0 and some of its weaker variations $A_1, A_2, \ldots, A_k$ and compares the new algorithms with an existing algorithm B. When B is better, the rank is $B \succ A_0 \succ \ldots \succ$

[1] Borda count is equivalent to rank sums r_j. Hence, whatever vote theory has to say about Borda count also applies to Friedman's test.

> A_k. When A_0 is better, the rank is $A_0 \succ A_1 \succ \ldots \succ A_k \succ B$. Therefore, the presence of $A_1, A_2, \ldots, A_k$ artificially increases the difference between A_0 and B.

But Ranked pairs method fulfills this criterion.

Finally, Friedman's test violates *Majority criterion* [3]:

> If one algorithm is the best on more than 50% of datasets, then that algorithm must win.

We have already observed a violation of this criterion in the though example with bagging versus boosting — even thought boosting was the best algorithm on the majority of the datasets, this fact alone did not guarantee boosting's victory. The violation of Majority criterion also implies a violation of *Condorcet criterion* [5]:

> If there is an algorithm which wins pairwise to each other algorithm, then that algorithm must win.

Which is, nevertheless, once again fulfilled with Ranked pairs method. However, just like in machine learning we have no free lunch theorem, Arrow's impossibility theorem [1] states that there is not a ranked vote method without a flaw. The flaw of all ranked vote methods, but dictatorship[2], that fulfill Condorcet criterium is that they fail *Consistency criterium* [18, Theorem 2]:

> If based on a set of datasets $\mathbb{A}$ an algorithm A is the best. And based on another set of datasets $\mathbb{B}$ an algorithm A is, again, the best. Then based on $\mathbb{A} \cup \mathbb{B}$ the algorithm A must be the best.

Notably, Friedman's test fulfills this criterium while Ranked pairs method fails this criterium. For convenience, a summary table with the list of discussed criteria is given in Table 1.

Table 1: Method compliance with criteria.

Criterium	Friedman's	Ranked pairs
IIA	✗	✗
LIIA	✗	✓
Independence of Clones	✗	✓
Majority	✗	✓
Condorcet	✗	✓
Consistency	✓	✗

4.3 Bayesian

Another option is to go parametric and replace the common assumption of normal distributions with Beta distributions, which are more appropriate for modeling of upper and bottom bounded measures [7, 12].

[2] In a dictatorship, the quality of an algorithm is determined on a single dataset

4.4 Continue Using Friedman's Test

Finally, there is the option of continuing using Friedman's test as before. In the numerical analysis in Section 3.2, we did not observe any *significant flip* in the ordering of the algorithms.

5 Conclusion

Contrary to our expectations, based on analysis of 179 classifiers on 121 datasets, Friedman's test appears to be fairly resistant to manipulation, where we add or remove irrelevant classifiers from the analysis. Therefore, we *cannot* recommend avoiding Friedman's test only because it violates Independence of Irrelevant Alternatives (IIA) criterium.

6 Acknowledgment

I would like to thank Adéla Chodounská for her help. We furthermore thank the anonymous reviewers, their comments helped to improve this paper. The reported research has been supported by the Grant Agency of the Czech Technical University in Prague (SGS17/210/OHK3/3T/18) and the Czech Science Foundation (GAČR 18-18080S).

References

[1] Kenneth J. Arrow. A Difficulty in the Concept of Social Welfare. *J. Polit. Econ.*, 58(4):328–346, 1950.

[2] Eric Bauer and Ron Kohavi. An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. *Mach. Learn.*, 36(1-2):105–139, 1999.

[3] Harry Beatty. Voting Rules and Coordination Problems. In *Methodol. Unity Sci.*, pages 155–189. Springer Netherlands, Dordrecht, 1973.

[4] Alessio Benavoli, Giorgio Corani, and Francesca Mangili. Should we really use post-hoc tests based on mean-ranks? *J. Mach. Learn. Res.*, 17:1–10, 2016.

[5] Duncan Black. On the Rationale of Group Decision-making. *J. Polit. Econ.*, 56(1):23–34, feb 1948.

[6] Leo Breiman. Bagging predictors. *Mach. Learn.*, 24(2):123–140, 1996.

[7] Giorgio Corani, Alessio Benavoli, Janez Demšar, Francesca Mangili, and Marco Zaffalon. Statistical comparison of classifiers through Bayesian hierarchical modelling. *Mach. Learn.*, 106(11):1817–1837, 2017.

[8] Nicholas J. Cox. Correlation with confidence, or Fisher's z revisited. *Stata J.*, 8(3):413–439, 2008.

[9] Janez Demšar. Statistical Comparisons of Classifiers over Multiple Data Sets. *J. Mach. Learn. Res.*, 7:1–30, 2006.

[10] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we Need Hundreds of Classifiers to Solve Real World Classification Problems? *J. Mach. Learn. Res.*, 15:3133–3181, 2014.

[11] Yoav Freund and Robert E. Schapire. Experiments with a New Boosting Algorithm. *Int. Conf. Mach. Learn.*, pages 148–156, 1996.

[12] John K. Kruschke. Bayesian data analysis. *Wiley Interdiscip. Rev. Cogn. Sci.*, 1(5):658–676, 2010.

[13] John W. Pratt. Remarks on Zeros and Ties in the Wilcoxon Signed Rank Procedures. *Am. Stat. Assoc.*, 54(287):655–667, 1959.

[14] Paramesh Ray. Independence of Irrelevant Alternatives. *Econometrica*, 41(5):987, sep 1973.

[15] Markus Schulze. A new monotonic and clone-independent single-winner election method. *Voting Matters*, (17):9–19, 2003.

[16] Thorwald N. Tideman. Independence of clones as a criterion for voting rules. *Soc. Choice Welfare*, 4(3):185–206, 1987.

[17] Bogdan Trawiński, Magdalena Smetek, Zbigniew Telec, and Tadeusz Lasota. Nonparametric statistical analysis for multiple comparison of machine learning regression algorithms. *Int. J. Appl. Math. Comput. Sci.*, 22(4):867–881, jan 2012.

[18] H. P. Young and A. Levenglick. A Consistent Extension of Condorcet's Election Principle. *SIAM J. Appl. Math.*, 35(2):285–300, sep 1978.

Appendix: Arrow's theorem

Arrow's theorem, once applied on algorithms and datasets, states that once we have 3 or more algorithms, a ranking method cannot fulfill all following "reasonable" properties:

Unrestricted domain The ranking method should be complete in that given a choice between algorithms A and B it should say whether A is preferred to B, or B is preferred to A or that there is indifference between A and B.

Transitivity The preferences should be transitive; i.e., if A is preferred to B and B is preferred to C then A is also preferred to C.

Non-dictatorship The outcome should not depend only upon a single dataset.

Weak Pareto Efficiency If algorithm A is better than algorithm B on all datasets, then A must rank above B.

Independence of Irrelevant Alternatives (IIA) If algorithm A is preferred to algorithm B out of the choice set $\{A,B\}$, introducing a third option X, expanding the choice set to $\{A,B,X\}$, must not make B preferable to A.

Appendix: Criteria

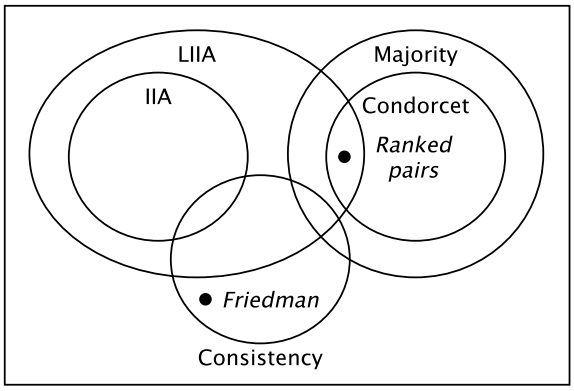


Figure 4: Venn diagram with the dependencies between the discussed vote theory criteria. IIA and consistency are inconceivable with the Condorcet criterion. Independence of clones is independent of other criteria and is not depicted. The diagram holds when following conditions from Arrow's theorem are satisfied: unrestricted domain, transitivity, non-dictatorship, 3 or more competing algorithms.

S. Krajči (ed.): ITAT 2018 Proceedings, pp. 64–71
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Jakub Repický, Martin Holeňa, and Zbyněk Pitra

Automated Selection of Covariance Function for Gaussian Process Surrogate Models

Jakub Repický[1,2] and Zbyněk Pitra[2,3] and Martin Holeňa[2]

[1] Faculty of Mathematics and Physics, Charles University in Prague
Malostranské nám. 25, 118 00 Prague 1, Czech Republic
[2] Institute of Computer Science, Czech Academy of Sciences
Pod Vodárenskou věží 2, 182 07 Prague 8, Czech Republic
{repicky,martin}@cs.cas.cz
[3] Faculty of Nuclear Sciences and Physical Engineering, CTU in Prague
Břehová 7, 115 19 Prague 1, Czech Republic

Abstract: Gaussian processes have a long tradition in model-based algorithms for black-box optimization, where a limited number of objective function evaluations are available. A principal choice in specifying a Gaussian process model is the choice of the covariance function, which largely embodies the prior assumptions about the modeled function. Several methods for learning the form of covariance function have been proposed. We report a work in progress in which the covariance function is selected from a fixed set. The goal of covariance function selection is to capture non-local properties of the objective function and derive a more accurate surrogate model. The model-selection algorithm is evaluated in connection with Doubly Trained Surrogate Covariance Matrix Adaptation Evolution Strategy on the Comparing Continuous Optimizers framework. Several estimates of predictive performance, including cross-validation and information criteria, are discussed. Focus is placed on information criteria suitable for nonparametric methods, and two of them are compared experimentally.

1 Introduction

The principle of continuous black-box optimization is finding extrema of real-parameter objective function analytical definition of which is not known. Such functions, often arising, e. g., in engineering design optimization or material science, can only be evaluated empirically or through simulations. Moreover, obtaining function values may be expensive and affected by noise. The goal of finding a global optimum is usually relaxed in favor of finding a good enough solution within as few objective function evaluations as possible.

Evolution strategies, stochastic population-based algorithms inspired by the process of natural evolution, present a popular approach to continuous black-box optimization. The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [10, 13] is based on adaptation of the key component of the mutation operator (the covariance matrix) according to the historical search steps. The CMA-ES is considered a state-of-the-art continuous black-box optimizer. Nevertheless, considerable improvements in terms of the number of fitness evaluations can be achieved by use of surrogate models, i. e., statistical or machine learning models of the fitness trained on data gathered during the optimization.

A variety of models for the CMA-ES has been investigated, including but not limited to quadratic approximations [14], ranking support vector machines [16], random forests [4] and Gaussian processes (GPs) [4, 19, 25].[1]

Gaussian process (GP) regression is a nonparametric method, meaning the data are assumed to be generated from an infinite-dimensional distribution, i. e., a distribution of functions. In black-box optimization, the distribution of function values conditioned on observed data can be used to derive a criterion for selecting most promising points for evaluation with the (expensive) fitness. As far as we know, the first optimization method utilizing uncertainty modeled by GPs is Bayesian optimization [17]. In this paper, we are going to build upon the more recent Adaptive Doubly Trained Surrogate CMA-ES (aDTS-CMA-ES), which uses a Gaussian process surrogate models for the CMA-ES, although our approach is directly applicable to Bayesian optimization as well.

A Gaussian process is fully specified by a mean function and a covariance function parametrized a by a small number of parameters. In order to distinguish parameters of the mean and covariance functions from the infinite-dimensional parameter vector – the vector of function values – they are referred to as hyperparameters. In statistical works, the mean and covariance functions are chosen by the statistician in a cycle of model building and model checking.

The goal of this work is to lay out a suitable method for learning the form of covariance function for Gaussian processes in black-box optimization with focus on criteria for evaluating candidate covariance functions. The main hypothesis behind this paper is that a GP with a composite form of its covariance function may result in a more accurate approximation of the objective function and, consequently, better performance of the model-assisted optimization algorithm.

[1] An experimental comparison of selected surrogate-assisted variants of the CMA-ES can be found in [3, 20].

Related Work Learning a composite expression of kernel functions for support vector machines by genetic programming was explored in [7].

Hierarchical kernel learning [2] and Additive Gaussian processes [6] are algorithms for determining kernels composed of lower-dimensional kernels.

The goal of Automatic Statistician project [15] is automatic statistical analysis of given data with output in natural language. The algorithm of structure discovery in GP models [5] is a greedy search in the space of composite covariance functions generated by operators of addition and multiplication recursively applied to basis covariance functions.

Up to our knowledge, structure discovery in GP surrogate models for continuous black-box optimization has not yet been investigated. As a first step towards this goal, we perform selection of the best GP model from a model population that we tried to design large enough to capture structure of typical continuous black-box function but still small enough for model selection to be computationaly feasible.

The paper is organized as follows. Section 2 presents ideas behind surrogate models in evolutionary optimization and aDTS-CMA-ES algorithm. Section 3 describes inference and learning in Gaussian process regression models. Section 4 presents the algorithm for selecting the best GP surrogate model. First results from an early stage of experimental evaluation are presented in Section 5. Section 6 concludes the paper.

2 Surrogate-Assisted Evolutionary Optimization

Evolutionary strategies are stochastic search algorithms based on maintaining a population of candidate solutions, usually encoded as real vectors. In each iteration (generation), a population of λ offsprings is generated from a population of μ parents by operators of recombination and mutation. The new population of parents is selected either from the union of offsprings and parents (*plus* selection), or, provided that $\mu \leq \lambda$, from the offsprings exclusively (*comma* selection).

2.1 CMA-ES

Mutation in evolutionary strategies is usually implemented by sampling from a Gaussian distribution, parameters of which play a crucial role in algorithms' convergence. The main idea behind the CMA-ES is self-adaptation of mutation parameters, especially of the covariance matrix. The CMA-ES repeatedly samples from $\mathcal{N}(\mathbf{m}, \sigma^2\mathbf{C})$ and updates parameters σ^2 (overall step-size), $\mathbf{m}$ (the mean) and $\mathbf{C}$ (the covariance matrix) so that likelihood of successful mutation steps increases under new parametrization.

Algorithm 1 aDTS-CMA-ES

Input: λ (population-size), y_{target} (target value), f (original fitness function), α (ratio of original-evaluated points), $\mathscr{C}$ (uncertainty criterion)

1: $\sigma, \mathbf{m}, \mathbf{C} \leftarrow$ CMA-ES initialize
2: $\mathscr{A} \leftarrow \emptyset$ {*archive initialization*}
3: **while** stopping conditions not met **do**
4: $\{\mathbf{x}_k\}_{k=1}^{\lambda} \sim \mathcal{N}\left(\mathbf{m}, \sigma^2\mathbf{C}\right)$ {*CMA-ES sampling*}
5: $f_{\mathscr{M}1} \leftarrow \text{trainModel}(\mathscr{A}, \sigma, \mathbf{m}, \mathbf{C})$ {*model training*}
6: $(\hat{\mathbf{y}}, \mathbf{s}^2) \leftarrow f_{\mathscr{M}1}([\mathbf{x}_1, \ldots, \mathbf{x}_\lambda])$ {*model evaluation*}
7: $\mathbf{X}_{\text{orig}} \leftarrow$select $\lceil \alpha\lambda \rceil$ best points accord. to $\mathscr{C}(\hat{\mathbf{y}}, \mathbf{s}^2)$
8: $\mathbf{y}_{\text{orig}} \leftarrow f(\mathbf{X}_{\text{orig}})$ {*original fitness evaluation*}
9: $\mathscr{A} = \mathscr{A} \cup \{(\mathbf{X}_{\text{orig}}, \mathbf{y}_{\text{orig}})\}$ {*archive update*}
10: $f_{\mathscr{M}2} \leftarrow \text{trainModel}(\mathscr{A}, \sigma, \mathbf{m}, \mathbf{C})$ {*model retrain*}
11: $\mathbf{y} \leftarrow f_{\mathscr{M}2}([\mathbf{x}_1, \ldots, \mathbf{x}_\lambda])$ {2^{nd} *model prediction*}
12: $(\mathbf{y})_i \leftarrow y_{\text{orig},i}$ for all original-evaluated $y_{\text{orig},i} \in \mathbf{y}_{\text{orig}}$
13: $\alpha \leftarrow \text{selfAdaptation}(\mathbf{y}, \hat{\mathbf{y}})$
14: $\sigma, \mathbf{m}, \mathbf{C} \leftarrow$ CMA-ES update
15: **end while**
16: $\mathbf{x}_{\text{res}} \leftarrow \mathbf{x}_k$ from $\mathscr{A}$ where y_k is minimal

Output: $\mathbf{x}_{\text{res}}$ (point with minimal y)

2.2 aDTS-CMA-ES

The aDTS-CMA-ES [3, 19, 21], utilizes a GP surrogate model to estimate the fitness of a fraction of the population. A pseudocode is given in Algorithm 1. The algorithm expects an uncertainty criterion $\mathscr{C}$ for choosing solutions for re-evaluation. In optimization based on Gaussian processes, such criteria are conveniently defined on the marginal GP posterior, which is a univariate Gaussian distribution. One of the most prominent uncertainty criteria is the probability of improvement, $\mathscr{C}_{\text{POI}}(\mathbf{x}; T) = Pr(f(\mathbf{x}) \leq T)$, i. e., the posterior probability that the function value at a candidate solution $\mathbf{x}$ improves on a chosen target T, typically set to the historically best fitness value.

The sampling in aDTS-CMA-ES is identical to that of CMA-ES. The surrogate model is trained twice per generation. The first model is trained on a data set, which naturally cannot contain any individuals from the current population. A fraction α of the population is selected according to $\mathscr{C}$, evaluated with the (expensive) fitness function and included into the archive of individuals with known fitness values. The model is retrained and used to predict the remainder of the population. The fraction α is adapted according to surrogate model performance.

3 Gaussian Processes

Let $\mathscr{X}$ be some input space of dimensionality D. Gaussian process with a mean function $\mu\colon \mathscr{X} \to \mathbb{R}$ and a covariance function $k\colon \mathscr{X} \times \mathscr{X} \to \mathbb{R}$, is a collection of random variables $(f(\mathbf{x}))_{\mathbf{x}\in\mathscr{X}}$ such that every finite-variate marginal $(f(\mathbf{x}_i))_{i=1}^N$ follows a multivariate Gaussian distribution $\mathcal{N}(\mu(X), K(X,X))$, where $\mu(X) = (\mu(\mathbf{x}_i))_{i=1}^N$ and

$K(X,X) = (k(\mathbf{x}_i,\mathbf{x}_j))_{i,j=1}^{N}$. Both μ and k are parameterized, but we omit their parameters for the sake of readability.

3.1 Inference

Let $\mathbf{y} = \{y_1,\ldots,y_N\}$ be N $i.i.d.$ observations at inputs $X = \{\mathbf{x}_1,\ldots,\mathbf{x}_N\}$. A model with Gaussian likelihood and GP prior is given by distributions $\mathbf{y}\,|\,\mathbf{f} \sim \mathcal{N}(\mathbf{f},\sigma_n^2 I_N)$ and $\mathbf{f}\,|\,X \sim \mathcal{N}(\mu(X),K(X,X))$. From now on, we assume $\mu = 0$. Deterministic non-zero mean functions can be used by simply substracting from $\mathbf{y}$ (see [22] for more on this). Let us denote by θ the vector of hyperparameters consisting of parameters of k and noise variance σ_n^2.

The marginal likelihood of hyperparameters θ is (see [22])

$$p(\mathbf{y}\,|\,X,\theta) = \int p(\mathbf{y}\,|\,X,f,\theta)p(f\,|\,\theta)df \tag{1}$$

$$= \varphi(\mathbf{y}\,|\,\mathbf{0},K(X,X)+\sigma_n I_N), \tag{2}$$

where φ denotes the normalized multivariate Gaussian density.

In the regression problem, we are interested in conditional distribution $\mathbf{f}_*\,|\,\mathbf{y},X,X_*,\theta$ for X_* a set of N_* test inputs. Since $[\mathbf{y}^T\mathbf{f}_*^T]^T\,|\,X,X_*,\theta$ follows a multivariate Gaussian distribution, the distribution of $\mathbf{f}_*\,|\,\mathbf{y},X,X_*,\theta$ is also a multivariate Gaussian, in particular

$$\mathbf{f}_* \sim \mathcal{N}(\hat{\mathbf{f}}_*,\mathrm{cov}(\mathbf{f}_*)), \text{ where} \tag{3}$$

$$\hat{\mathbf{f}}_* = K(X_*,X)[K(X,X)+\sigma_n^2 I_N]^{-1}\mathbf{y} \tag{4}$$

$$\mathrm{cov}(\mathbf{f}_*) = K(X_*,X_*) - K(X_*,X)[K(X,X)+\sigma_n I_N]^{-1}K(X,X_*) \tag{5}$$

3.2 Hierarchical Model

When the covariance function family is given, model selection for GP regression is usually performed by maximum marginal likelihood estimate $\hat{\theta}_{\mathrm{ML}} = \arg\max_\theta \log p(y\,|\,X,\theta)$, which is a non-convex optimization problem. Computation of log marginal likelihood takes $\mathcal{O}(N^3)$ time due to a Cholesky decomposition of covariance matrix $K(X,X)$.

From a Bayesian perspective, especially if the number of hyperparameters is large or if N is small, it might be more appropriate to do inference with the marginal posterior distribution of hyperparameters

$$p(\theta\,|\,X,\mathbf{y}) = \frac{p(\mathbf{y}\,|\,X,\theta)p(\theta)}{p(\mathbf{y}\,|\,X)}, \tag{6}$$

where $p(\mathbf{y}\,|\,X,\theta)$ is the marginal likelihood (1), now playing the role of the likelihood, and $p(\theta)$ is a hyper-prior. Simulations from $p(\theta\,|\,X,\mathbf{y})$ can be obtained by Bayesian computation methods, such as Markov chain Monte Carlo.

Uncertainty criteria in Algorithm 1 can thus incorporate uncertainty of hyperparameter estimation in addition to uncertainty about functions. In the current stage of research, we compute the prediction conditioned on a Bayes estimate $\theta_{\mathrm{Bayes}} = \mathrm{median}(\{\theta_s, s = 1,\ldots,S\})$, i. e., the median of the posterior sample.

4 Model Selection

If the probability of the true fitness function under GP prior is low, the performance of the model will be poor. For example, a GP with a neural network covariance fits data from a jump function better compared to a GP with a squared exponential [22] (more on covariance functions in Subsection 4.1). Searching over GP models with different covariances thus can be viewed as an automated construction of suitable priors. We select the model from a finite set according to a criterion of predictive performance, since this approach can easily be embedded into a combinatorial search algorithm, such as in [5]. GPs can represent random functions. The finite population of models included in our approach is described in Subsection 4.1. Some important classes of functions, such as linear and quadratic functions, neural networks and additive functions, are represented.

4.1 Model Population

The set of candidate GP models is shown in Figure 1. All models have zero mean.

A covariance function $k(\mathbf{x},\mathbf{x}')$ is stationary if it is a function of a distance $\|\mathbf{x}-\mathbf{x}'\|$. The squared exponential (SE) [22] is a stationary covariance function that leads to smooth processes [22].

A neural network (NN) covariance is a covariance of a GP induced by a Gaussian prior on weights of an infinitely wide neural network [18].

A dot product with a bias constant term models linear functions. The quadratic covariance is such a linear covariance squared. GPs with these covariances lead to Bayesian variants of linear and quadratic regression, respectively.

Additive covariance functions [6] are sums of lower dimensional components. We include an additive covariance function with a single degree of interaction – a superposition of one-dimensional squared exponentials.

Finally, we consider two cases of composite covariance functions: a sum of a squared exponential and a neural network; and a sum of a squared exponential and a quadratic.

4.2 Performance Criteria

We would like to select the surrogate model based on an estimation of out-of-sample predictive accuracy.

An attractive estimate of the out-of-sample predictive accuracy is cross-validation based on some partitioning of the data set into multiple data sets called folds. However, choosing among multiple GP models by cross-validation in each generation of the evolutionary optimization can

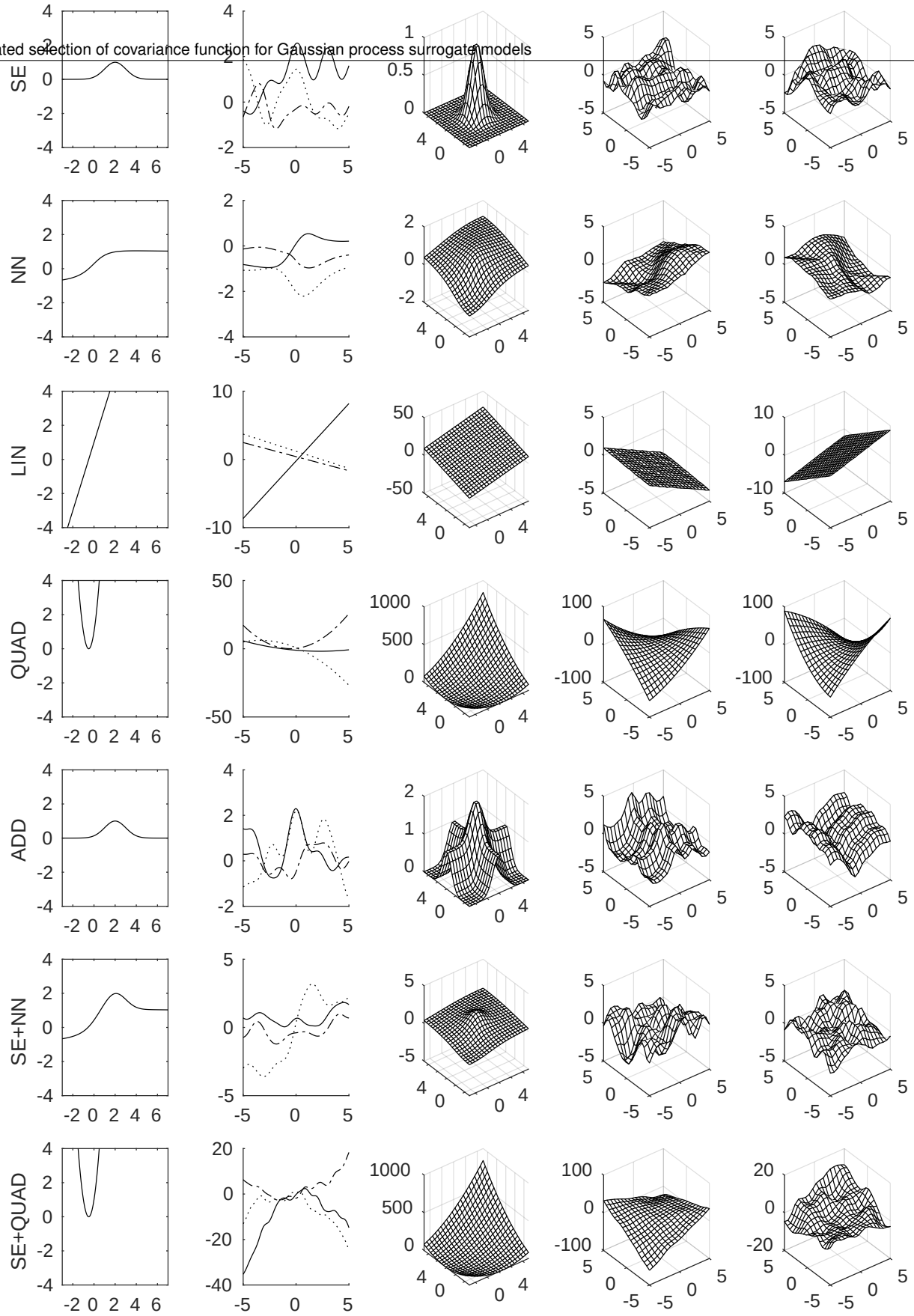


Figure 1: Rows: Used covariance functions. Columns 1–2: The covariance function on $\mathbb{R}$ centered at point 2 (Col. 1) and three independent samples from the GP (Col. 2). Columns 3–5: The covariance function on $\mathbb{R}^2$ centered at $[2\ 2]^T$ (Col. 3) and two independent samples from the GP (Col. 4 and 5).

be considered prohibitive from the computational perspective.

In the remainder of this subsection we follow the exposition of model comparison from Bayesian perspective given in [8]. We denote by q the true distribution from which data $\mathbf{y}$ are sampled and we suppress conditioning on X for simplicity.

A general measure of fit of a probabilistic model $\mathbf{y}$ to data is the log likelihood or log predictive density $\log p(\mathbf{y}\,|\,\theta) = \log\prod_{i=1}^{N} p(y_i\,|\,\theta)$. The quantity $-2\log p(y\,|\,\theta)$ is called deviance.

Akaike information criterion (AIC) [1] and related Bayes information criterion (BIC) [23] are based on the expected log predictive density conditioned on a maximum likelihood estimate $\hat{\theta}_{\mathrm{ML}}$,

$$\mathrm{elpd}_{\hat{\theta}} = E_q(\log p(\tilde{\mathbf{y}}\,|\,\hat{\theta}_{\mathrm{ML}})), \tag{7}$$

where the expectation is taken over all possible data sets $\tilde{\mathbf{y}}$. Since expectation (7) cannot be computed exactly, it is estimated from sample $\mathbf{y}$. The AIC and BIC compensate for the bias towards overfitting by substracting a correction term, the number of parameters n_θ and $\frac{1}{2}n_\theta \log N$, respectively.

For hierarchical Bayesian models, such as (6), it is not always entirely clear, what the parameters of the model are, since the likelihood can factorize in different ways. The deviance information criterion (DIC) [24] is still based on deviance, conditioned on a Bayes estimate $\hat{\theta}_{\mathrm{Bayes}}$, but the effective number of parameters p_{DIC} depends on data. We define the DIC for the marginal likelihood (1), focusing on hyperparameters θ, although it could be defined for the likelihood $p(\mathbf{y}\,|\,f,\theta)$, focusing on both f and θ.

We use the following definition of the effective number of parameters (see [8]):

$$p_{\mathrm{DIC}} = 2\mathrm{var}_{\mathrm{post}}(\log p(\mathbf{y}\,|\,\theta)),$$

which can be estimated by the sample variance of a posterior sample. Using the effective number of parameters, the DIC is

$$DIC = -2\log p(\mathbf{y}\,|\,\hat{\theta}_{\mathrm{Bayes}}) + 2p_{\mathrm{DIC}}.$$

A probabilistic model is called regular if its parameters are identifiable and its Fisher information matrix is positive definite for all parameter values. The model is called singular otherwise. The information criteria defined above assume regularity. The Widely applicable information (WAIC) [26] works also for singular models. The WAIC is based on estimation of the expected log *pointwise* predictive density

$$\begin{aligned}\mathrm{elppd} &= \sum_{i=1}^{N} E_q(\log p_{\mathrm{post}}(\tilde{y}_i)) \\ &= \sum_{i=1}^{N} E_q(\log \int p(\tilde{y}_i\,|\,\mathbf{y},\theta)p(\theta\,|\,\mathbf{y})d\theta.\end{aligned}$$

The estimation of elppd from the sample is biased, so again, an effective number of parameters must be added as a correction. We use the following definition of the WAIC (see [8]):

$$WAIC = -\sum_{i=1}^{N}\log p_{\mathrm{post}}(y_i) + \sum_{i=1}^{N}\mathrm{var}_{\mathrm{post}}(\log p(y_i\,|\,\theta)),$$

that is the negative log pointwise predictive density corrected for bias by pointwise posterior variance of log predictive density.

The pointwise predictive density $p_{\mathrm{post}}(y_i\,|\,\mathbf{y},\theta)$ for the GP model (1) is computed by integrating Gaussian likelihood over the marginal posterior GP at i^{th} training point:

$$\begin{aligned}p(y_i\,|\,\mathbf{y},\theta) &= \int p(y_i\,|\,\mathbf{y},f_i,\theta)p(f_i\,|\,\mathbf{y},\theta)\,df_i \\ &= \varphi(y_i\,|\,\hat{f}_i,\sigma_n^2 + \mathrm{var}(f_i)),\end{aligned}$$

where φ denotes the Gaussian density and $\hat{f}_i, \mathrm{var}(f_i)$ are as in (3).

5 Experimental Evaluation

In this section, we describe preliminary experimental evaluation procedure of aDTS-CMA-ES that uses a GP model with an automated selection of covariance function. Since GPs are a nonparametric model, we opt for the WAIC, which require a sample from distribution (6). We use Metropolis-Hastings MCMC with an adaptive proposal distribution [9] [2].

Algorithm 1 is updated in the following way: [3]

1. In steps (5) and (10), all GPs from Figure 1 are trained.
2. The predictive accuracy of all models is evaluated using the WAIC (4.2). The DIC (4.2) is also computed for information, but not taken into account.
3. The model with the lowest WAIC is used for prediction (steps (6) and (11)).

The hyper-priors are chosen as follows: log-normal with mean $\log(0.01)$ and variance 2 for σ_n^2; and log-$t_{\nu=4}$ with mean 0 for all other hyperpameters.

5.1 Setup

The proposed algorithm implemented in MATLAB is evaluated on the noiseless testbed of the COCO/BBOB (Comparing Continuous Optimizers / Black-Box Optimization Benchmarking) framework [11,12] and compared with the GP-based aDTS-CMA-ES and the CMA-ES itself.

[2] Using MATLAB implementation available at `http://helios.fmi.fi/~lainema/dram/`

[3] The sources are available at `https://github.com/repjak/surrogate-cmaes/tree/modelsel`

The testbed consists of 24 functions, each defined everywhere on $\mathbb{R}^D$ with the optimum in $[-5,5]^D$ for all dimensionalities $D \geq 2$. Each test function has multiple instances which are derived by various transformations of input space or f-space. We run the algorithm on 5 instances $(1 \ldots, 5)$ as opposed to 15 recommended instances for the reason of increased computational demands of the modified algorithm. For the same reason, only functions of 10 variables ($10D$) are considered.

If not stated otherwise, all settings of the aDTS-CMA-ES are as recommended in [3].

The CMA-ES results in BBOB format were downloaded from the BBOB 2010 workshop archive [4].

5.2 Results

Figure 2 gives the scaled best-achieved logarithms $\Delta_f^{\log}$ of median distances to the functions optimum for the respective number of function evaluations per dimension (FE/D). Medians and the 1st and 3rd quartiles are calculated from 5 independent instances in case of the algorithm with covariance selection according to the WAIC and from 15 independent instances otherwise. We observe that in most cases, the WAIC-based algorithm mostly barely outperforms the pure CMA-ES, which suggests the chosen model is generally weak and the adaptivity mechanism basically turns off using the surrogate model. The functions where the WAIC variant outperforms the aDTS-CMA-ES (f21 and f22) are multi-modal and the interquartile range is large.

In order to compare the considered information criteria, we calculate the rank of each model under both WAIC and DIC. Table 1 summarizes the average ranks over all model selections performed on each benchmark function. We observe that the DIC often prefers the additive model, while the WAIC is more balanced in this respect. Surprisingly the linear kernel has been very rarely selected even on the linear function (f5) under both information criteria. A similar observation holds for the quadratic kernel and the quadratic functions (f1, f2).

6 Conclusion & Further Work

In this paper, we presented an algorithm for selecting a GP kernel using Bayesian model comparison techniques. Preliminary experiments for the model selection plugged into the aDTS-CMA-ES algorithm were conducted on the COCO/BBOB testbed. Due to the small number of experiments performed so far, it is difficult to draw any serious conclusions. The first obtained results may indicate improper convergence of the MCMC sampler or that more sophisticated covariance functions may be needed.

One direction of future research, beside analyzing and repairing aforementioned deficiencies, is an extension of the proposed algorithm into a combinatorial search over kernels in flavor of [5,7], which is challenging due to computational costs related to the need of repeated surrogate model retraining.

One possible direction of research is a *co-evolution* of a population of covariance functions alongside the population of candidate solutions to the black-box objective function. Other related research area is applying surrogate modeling to high-dimensional problems using algorithms for variable selection via multiple kernel learning [2, 6].

[4] `http://coco.gforge.inria.fr/data-archive/bbob/2010/`

Acknowledgements

This research was supported by SVV project number 260 453 and the Czech Science Foundation grants No. 17-01251. Further, access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum provided under the programme "Projects of Large Research, Development, and Innovations Infrastructures" (CESNET LM2015042), is greatly appreciated.

References

[1] H. Akaike. *Information Theory and an Extension of the Maximum Likelihood Principle*, pages 199–213. Springer New York, 1973.

[2] F. Bach. High-dimensional non-linear variable selection through hierarchical kernel learning, 2009.

[3] L. Bajer. *Model-based evolutionary optimization methods*. PhD thesis, Faculty of Mathematics and Physics, Charles University in Prague, Prague, 2018.

[4] L. Bajer, Z. Pitra, and M. Holeňa. Benchmarking Gaussian processes and random forests surrogate models on the BBOB noiseless testbed. In *Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference - GECCO Companion '15*. Association for Computing Machinery (ACM), 2015.

[5] D. Duvenaud, J. Lloyd, R. Grosse, J. Tenenbaum, and G. Zoubin. Structure discovery in nonparametric regression through compositional kernel search. In S. Dasgupta and D. McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1166–1174, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[6] D. K. Duvenaud, H. Nickisch, and C. E. Rasmussen. Additive gaussian processes. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 226–234. Curran Associates, Inc., 2011.

[7] C. Gagné, M. Schoenauer, M. Sebag, and M. Tomassini. Genetic programming for kernel-based learning with co-evolving subsets selection. In *PARALLEL PROBLEM SOLVING FROM NATURE, REYKJAVIK, LNCS*, pages 1008–1017. Springer Verlag, 2006.

[8] A. Gelman, J. Hwang, and A. Vehtari. Understanding predictive information criteria for bayesian models. *Statistics and Computing*, 24(6):997–1016, Nov. 2014.

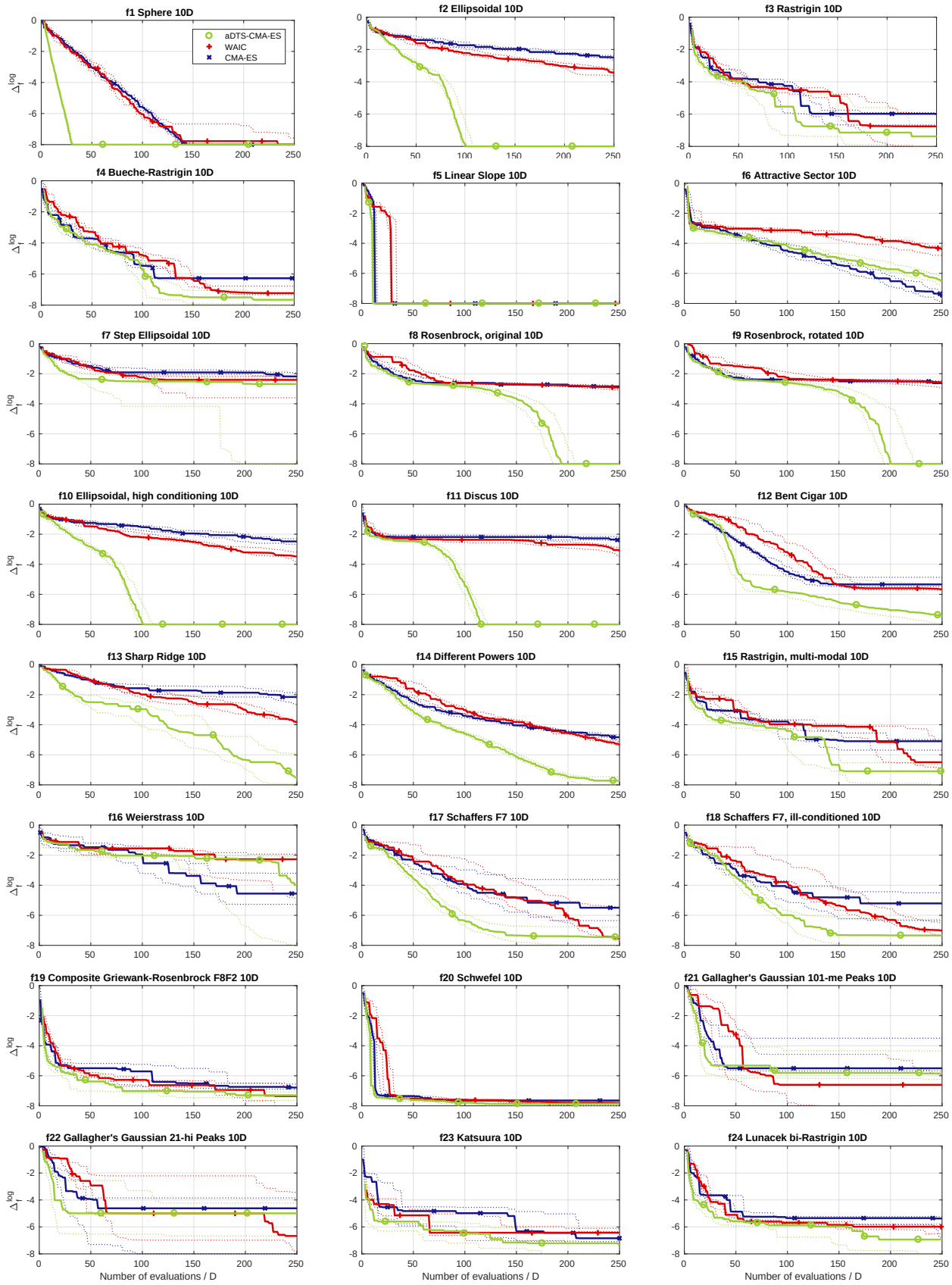


Figure 2: Medians (solid) and 1st/3rd quartiles (dotted) of the distances to the optima of 24 COCO/BBOB benchmarks in 10D for algorithms aDTS-CMA-ES (green), aDTS-CMA-ES with WAIC-based model selection (red) and CMA-ES (blue). The medians and quartiles for WAIC variant were calculated from 5 independent instances. In all other cases, 15 indepenent instances were used. Distances to optima are shown in the $\log_{10}$ scale.

Table 1: Average model ranks in 10D for each predictive performance criterion. The lowest value in bold.

Criterion	WAIC							DIC						
Model	SE	NN	LIN	QUAD	ADD	SE+NN	SE+LIN	SE	NN	LIN	QUAD	ADD	SE+NN	SE+LIN
f1	3.64	4.01	6.94	4.17	**2.73**	3.25	3.27	5.57	4.68	6.65	3.11	**1.55**	4.07	2.37
f2	3.07	3.05	6.96	5.41	4.35	**2.48**	2.67	5.38	4.85	6.78	3.99	**1.45**	3.71	1.84
f3	2.94	3.20	6.75	5.99	4.14	**2.45**	2.53	4.37	4.63	6.74	5.40	**1.40**	3.11	2.36
f4	3.00	3.20	6.87	5.73	4.11	**2.52**	2.57	4.49	4.67	6.76	5.21	**1.30**	3.27	2.30
f5	3.69	3.60	6.78	4.41	**2.69**	3.28	3.56	6.75	4.16	4.71	3.25	3.48	3.00	**2.66**
f6	3.03	3.09	6.99	5.95	3.99	**2.46**	2.50	4.18	4.62	6.92	5.92	1.86	2.80	**1.69**
f7	3.15	3.09	6.92	5.86	3.91	**2.49**	2.58	4.35	4.87	6.90	5.37	**1.54**	2.99	1.98
f8	2.83	3.12	6.97	5.76	4.15	**2.50**	2.66	4.78	4.57	6.83	5.21	**1.32**	3.27	2.02
f9	2.86	3.14	6.97	5.69	4.14	2.63	**2.57**	4.86	4.57	6.79	5.05	**1.32**	3.37	2.04
f10	3.00	3.16	6.96	5.43	4.31	**2.52**	2.62	5.39	4.82	6.76	3.95	**1.53**	3.79	1.75
f11	3.01	3.09	6.97	5.53	4.24	**2.57**	2.60	5.21	5.17	6.80	3.66	2.14	3.87	**1.16**
f12	3.16	3.28	6.93	4.37	4.96	**2.64**	2.66	5.46	4.65	6.66	3.57	**1.46**	3.94	2.27
f13	2.93	2.98	6.98	5.83	4.42	**2.37**	2.50	4.87	4.47	6.88	5.23	**1.24**	3.06	2.25
f14	3.29	2.96	7.00	5.90	3.63	**2.59**	2.64	4.26	4.84	6.99	5.64	**1.43**	3.04	1.81
f15	2.86	3.28	6.73	5.95	4.13	2.58	**2.47**	4.25	4.69	6.80	5.58	**1.48**	2.92	2.28
f16	2.53	3.59	6.46	6.50	4.16	2.41	**2.36**	3.69	4.69	6.78	6.10	**1.80**	2.44	2.51
f17	3.25	3.05	6.86	6.09	3.52	**2.58**	2.65	4.03	4.69	6.90	5.95	**1.36**	2.80	2.26
f18	3.17	3.05	6.88	6.10	3.65	**2.55**	2.60	3.98	4.70	6.88	6.00	**1.42**	2.74	2.28
f19	2.60	3.52	6.42	6.54	4.20	**2.32**	2.39	3.69	4.67	6.75	6.13	**1.45**	2.52	2.78
f20	2.98	3.37	6.94	5.55	4.01	2.62	**2.51**	4.46	4.73	6.81	5.22	**1.30**	3.26	2.21
f21	3.14	3.21	6.87	5.07	4.61	2.59	**2.51**	4.91	4.65	6.76	4.56	**1.34**	3.47	2.31
f22	3.11	3.22	6.88	5.00	4.57	**2.58**	2.63	5.06	4.60	6.73	4.31	**1.41**	3.51	2.37
f23	2.47	3.77	6.35	6.58	4.28	2.34	**2.21**	3.50	4.76	6.68	6.15	**1.74**	2.47	2.70
f24	2.73	3.53	6.44	6.51	4.06	2.38	**2.34**	3.76	4.68	6.77	6.11	**1.42**	2.53	2.73

[9] H. Haario, M. Laine, A. Mira, and E. Saksman. Dram: Efficient adaptive mcmc. *Statistics and Computing*, 16(4):339–354, Dec 2006.

[10] N. Hansen. *The CMA evolution strategy: a comparing review*, pages 75–102. Springer, Berlin, Heidelberg, 2006.

[11] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter Black-Box Optimization Benchmarking 2009: Noiseless functions definitions. Technical report, INRIA, 2009, updated 2010.

[12] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter Black-Box Optimization Benchmarking 2012: Experimental setup. Technical report, INRIA, 2012.

[13] N. Hansen and A. Ostermeier. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation*, 9(2):159–195, June 2001.

[14] S. Kern, N. Hansen, and P. Koumoutsakos. *Local Metamodels for Optimization Using Evolution Strategies*, pages 939–948. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[15] J. R. Lloyd, D. Duvenaud, R. Grosse, J. B. Tenenbaum, and Z. Ghahramani. Automatic construction and natural-language description of nonparametric regression models. *CoRR*, abs/1402.4304, Apr. 2014.

[16] I. Loshchilov, M. Schoenauer, and M. Sebag. Intensive surrogate model exploitation in self-adaptive surrogate-assisted cma-es (saacm-es). In *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference - GECCO '13*. ACM Press, 2013.

[17] J. Močkus. On bayesian methods for seeking the extremum. In *Proceedings of the IFIP Technical Conference*, London, UK, 1974. Springer-Verlag.

[18] R. M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996.

[19] Z. Pitra, L. Bajer, and M. Holeňa. Doubly trained evolution control for the surrogate CMA-ES. In *Parallel Problem Solving from Nature – PPSN XIV*, pages 59–68. Springer International Publishing, 2016.

[20] Z. Pitra, L. Bajer, J. Repický, and M. Holeňa. Overview of surrogate-model versions of covariance matrix adaptation evolution strategy. In *Proceedings of the Genetic and Evolutionary Computation Conference 2017, Berlin, Germany, July 15–19, 2017 (GECCO '17)*. ACM, July 2017.

[21] Z. Pitra, L. Bajer, J. Repický, and M. Holeňa. Adaptive Doubly Trained Evolution Control for the Covariance Matrix Adaptation Evolution Strategy. In *ITAT 2017: Information Technologies—Applications and Theory*, Martin, Sept. 2017. CreateSpace Independent Publ. Platform.

[22] C. E. Rassmusen and C. K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning series. MIT Press, 2006.

[23] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[24] D. Spiegelhalter, N. Best, B. Carlin, and A. Van Der Linde. Bayesian measures of model complexity and fit. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 64(4):583–616, 12 2002.

[25] H. Ulmer, F. Streichert, and A. Zell. Evolution strategies assisted by Gaussian processes with improved pre-selection criterion. In *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.*, pages 692–699. Institute of Electrical and Electronics Engineers (IEEE), 2003.

[26] S. Watanabe. Asymptotic equivalence of bayes cross validation and widely applicable information criterion in singular learning theory. *J. Mach. Learn. Res.*, 11:3571–3594, Dec. 2010.

S. Krajči (ed.): ITAT 2018 Proceedings, pp. 72–79
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Zbyněk Pitra, Jakub Repický, and Martin Holeňa

Boosted Regression Forest for the Doubly Trained Surrogate Covariance Matrix Adaptation Evolution Strategy

Zbyněk Pitra[1,2], Jakub Repický[1,3], and Martin Holeňa[1]

[1] Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2, 182 07 Prague 8, Czech Republic
{pitra, repicky, holena}@cs.cas.cz
[2] Faculty of Nuclear Sciences and Physical Engineering, CTU in Prague
Břehová 7, 115 19 Prague 1, Czech Republic
[3] Faculty of Mathematics and Physics, Charles University in Prague
Malostranské nám. 25, 118 00 Prague 1, Czech Republic

Abstract: Many real-world problems belong to the area of continuous black-box optimization, where evolutionary optimizers have become very popular inspite of the fact that such optimizers require a great amount of real-world fitness function evaluations, which can be very expensive or time-consuming. Hence, regression surrogate models are often utilized to evaluate some points instead of the fitness function. The Doubly Trained Surrogate Covariance Matrix Adaptation Evolution Strategy (DTS-CMA-ES) is a surrogate-assisted version of the state-of-the-art continuous black-box optimizer CMA-ES using Gausssian processes as a surrogate model to predict the whole distribution of the fitness function. In this paper, the DTS-CMA-ES is studied in connection with the boosted regression forest, another regression model capable to estimate the distribution. Results of testing regression forest and Gaussian processes, the former in 20 different settings, as a surrogate models in the DTS-CMA-ES on the set of noiseless benchmarks are reported.

1 Introduction

Real-world problems can be very costly in terms of various resources, most often money and time. An important kind of such problems are optimization tasks in which the objective function cannot be expressed mathematically, but has to be evaluated empirically, through measurements, experiments, or simulations. Such optimization tasks are called *black-box*. *Evolutionary algorithms* have become very successful in this optimization field. In case of limited resources, the number of empirical evaluations necessary to achieve the target distance to the optimal value by the optimization algorithm should be as small as possible.

Nowadays, the *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES) [12] is considered to be the state-of-the-art algorithm for continuous black-box optimization. On the other hand, the CMA-ES can require many function evaluations to achieve the target distance from the optimum in problems where the fitness function is expensive. Therefore, several surrogate-assisted versions of the CMA-ES have been presented during the last decade (an overview can be found in [2, 20]). Such CMA-ES based algorithms save expensive evaluations through utilization of a regression surrogate model instead of the original function on a selected part of the current population.

The local meta-model CMA-ES (lmm-CMA-ES) was proposed in [16] and later improved in [1]. The algorithm constructs local quadratic surrogate models and controls changes in population ranking after each evaluation of the original fitness.

In [18], the *Doubly Trained Surrogate CMA-ES* (DTS-CMA-ES) was presented. It utilizes the ability of *Gaussian Processes* (GPs) [21] to estimate the whole distribution of fitness function values to select most promising points to be evaluated by the original fitness. The DTS-CMA-ES was also tested [19] in a version where metric GPs were replaced by ordinal GPs inspired by the fact that the CMA-ES is invariant with respect to order preserving transformations. However, up to our knowledge, there has been no research into combining the DTS-CMA-ES with the surrogate model capable to predict the whole probability distribution of fitness values, where the model was not based on GPs. The ensembles of regression trees [4] are also able to estimate the whole distribution of values. They have been already utilized as surrogate models for the CMA-ES in [3] using different evolution control strategy than the DTS-CMA-ES.

In this paper, we use ensembles of regression trees as surrogate models in the DTS-CMA-ES algorithm. Due to an increasing popularity of gradient boosting [10], we train the ensembles of regression trees, i. e., *regression forests* (RFs), using such strategy. Up to our knowledge, this is the first time the boosted RF regression is utilized for surrogate modeling in the DTS-CMA-ES context. Therefore, we investigate also the suitability of several different settings of the employed regression method to this end. We experimentaly compare the original DTS-CMA-ES with a new version using RFs together with the original CMA-ES on the noiseless part of the *Comparing-Continuous-Optimizers* (COCO) platform [13, 14] in the expensive settings with the limited budget of fitness evaluations.

The rest of the paper is organized as follows. Section 2 describes the DTS-CMA-ES algorithm. Section 3 gives a short introduction into gradient boosting and regression

Algorithm 1 DTS-CMA-ES [18]

Input: λ (population-size), y_{target} (target value), f (original fitness function), n_{orig} (number of original-evaluated points), $\mathcal{C}$ (uncertainty criterion)

$\sigma, \mathbf{m}, \mathbf{C} \leftarrow$ CMA-ES initialize
$\mathcal{A} \leftarrow \varnothing$ {*archive initialization*}
while $\min\{y|(\mathbf{x},y) \in \mathcal{A}\} > y_{\text{target}}$ **do**
  $\{\mathbf{x}_k\}_{k=1}^{\lambda} \sim \mathcal{N}\left(\mathbf{m}, \sigma^2\mathbf{C}\right)$ {*CMA-ES sampling*}
  $f_{\mathcal{M}1} \leftarrow \text{trainModel}(\mathcal{A}, \sigma, \mathbf{m}, \mathbf{C})$ {*model training*}
  $(\hat{y}, \mathbf{s}^2) \leftarrow f_{\mathcal{M}1}([\mathbf{x}_1, \ldots, \mathbf{x}_\lambda])$ {*model evaluation*}
  $\mathbf{X}_{\text{orig}} \leftarrow$ select n_{orig} best points according to $\mathcal{C}(\hat{y}, \mathbf{s}^2)$
  $\mathbf{y}_{\text{orig}} \leftarrow f(\mathbf{X}_{\text{orig}})$ {*original fitness evaluation*}
  $\mathcal{A} = \mathcal{A} \cup \{(\mathbf{X}_{\text{orig}}, \mathbf{y}_{\text{orig}})\}$ {*archive update*}
  $f_{\mathcal{M}2} \leftarrow \text{trainModel}(\mathcal{A}, \sigma, \mathbf{m}, \mathbf{C})$ {*model retrain*}
  $\mathbf{y} \leftarrow f_{\mathcal{M}2}([\mathbf{x}_1, \ldots, \mathbf{x}_\lambda])$ {2^{nd} *model prediction*}
  $(\mathbf{y})_k \leftarrow y_{\text{orig},i}$ for all original-evaluated $y_{\text{orig},i} \in \mathbf{y}_{\text{orig}}$
  $\sigma, \mathbf{m}, \mathbf{C} \leftarrow$ CMA-ES update $(\mathbf{y}, \sigma, \mathbf{m}, \mathbf{C})$
end while
$\mathbf{x}_{\text{res}} \leftarrow \mathbf{x}_k$ from $\mathcal{A}$ where y_k is minimal

Output: $\mathbf{x}_{\text{res}}$ (point with minimal y)

tree algorithms. Section 4 contains experimental setup and results. Section 5 concludes the paper and discusses future research.

2 Doubly Trained Surrogate CMA-ES

The DTS-CMA-ES, introduced in [18], is a modification of the CMA-ES, where the ability of GPs to estimate the whole distribution of fitness function is utilized to select the most promising points out of the sampled population. The points selected using some uncertainty criterion $\mathcal{C}$ are evaluated with the original fitness function f and included into the set of points employed for the GP model retraining. The remaining points from the population are reevaluated using the retrained GP model. The core CMA-ES parameters (σ, $\mathbf{m}$, $\mathbf{C}$, etc.) are computed according to the original CMA-ES algorithm. The DTS-CMA-ES pseudocode is shown in Algorithm 1.

The only models capable to predict the whole fitness distribution used so far in connection with the DTS-CMA-ES were based on GPs.

3 Boosted regression forest

Regression forest [5] is an ensemble of regression decision trees [4]. In the last decade, the *gradient tree boosting* [10] has become very popular and successful technique for forest learning. Therefore, we will focus only on this method.

3.1 Gradient boosted regression trees

Let's consider binary regression trees, where each observation $\mathbf{x} = (x_1, x_2, \ldots, x_D) \in \mathbb{R}^D$ passes through a series of binary split functions s associated with internal nodes and arrives in the leaf node containing a real-valued constant trained to be the prediction of an associated function value y. A binary split function determines whether $\mathbf{x}$ proceeds to its left or right child of the respective node.

The gradient boosted forest has to be trained in an additive manner. Let $\hat{y}_i^{(t)}$ be the prediction of the i-th point of the t-th tree. The t-th tree f_t is obtained in the t-th iteration of the boosting algorithm through optimization of the following regularized objective function:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{N} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)\right) + \Omega(f_t), \qquad (1)$$

$$\text{where} \quad \Omega(f) = \gamma T_f + \frac{1}{2}\lambda \left\|w_f\right\|^2,$$

l is a differentiable convex loss function $l : \mathbb{R}^2 \to \mathbb{R}$, T_f is the number of leaves in a tree f, and w_f are weights of its individual leaves. The regularization term Ω is used to control model complexity through penalization constants $\gamma \geq 0$ and $\lambda \geq 0$.

The tree growing starts with one node (root) and a set of all input data. Individual branches are then recursively added according to the gain of split considering splitting data $\mathcal{S}_{L+R}$ into sets $\mathcal{S}_L$ (left branch set) and $\mathcal{S}_R$ (right branch set). The gain can be derived using (1) as follows (see [7] for details):

$$\mathcal{L}_{\text{split}} = \frac{1}{2}\left[r(\mathcal{S}_L) + r(\mathcal{S}_R) - r(\mathcal{S}_{L+R})\right] - \gamma,$$
$$r(\mathcal{S}) = \frac{\left(\sum_{y\in\mathcal{S}} g(y)\right)^2}{\sum_{y\in\mathcal{S}} h(y) + \lambda}, \qquad (2)$$

where $g(y) = \partial_{\hat{y}^{(t-1)}} l(y, \hat{y}^{(t-1)})$ and $h(y) = \partial^2_{\hat{y}^{(t-1)}} l(y, \hat{y}^{(t-1)})$ are the first and second order derivatives of the loss function.

The tree growing is stopped when one of the user-defined conditions is satisfied, e. g., the tree reaches the maximum number of levels, or no node can be split without dropping the number of points in any node under the allowed minimum.

The overall boosted forest prediction is obtained through averaging individual tree predictions, where each leaf j in a t-th tree has weight

$$w_j^{(t)} = -\frac{\sum_{y\in\mathcal{S}_j} g(y)}{\sum_{y\in\mathcal{S}_j} h(y) + \lambda}, \qquad (3)$$

where S_j is the set of all training inputs that end in the leaf j. As a prevention of overfitting, the random subsampling of input features or input points can be employed.

3.2 Split algorithms

The decision split function s can be found through numerous algorithms developed since the original CART [4]. In the following paragraphs, we survey some of them.

Traditional CART [4] are based on searching *axis-parallel* hyperplanes. To find the splitting hyperplane, the value of each training point $\mathbf{x} = (x^{(1)}, \ldots, x^{(D)})$ in dimension $\mathbf{x}^{(d)}$, $d \in \{1, \ldots, D\}$ is considered as a threshold for the dimension d defining the candidate hyperplane. The full-search through all dimensions is done and the splitting hyperplane with the highest gain is selected.

In SECRET [9], the expectation-maximization algorithm for *Gaussian mixtures* is utilized to find two clusters and the regression task is transformed into classification based on assignments of points to these clusters. Splitting oblique hyperplane is provided through linear or quadratic discriminant analysis.

Deterministic *hill-climbing* with effective randomization is employed to find a most suitable linear hyperplane in the algorithm OC1 [17]. Split-finding starts with a random hyperplane or with a good axis-parallel hyperplane found similarly to CART. Then the hyperplane's direction is deterministically perturbated in each axis to maximize the split gain. Once no improvement is possible, a number of random jumps is performed as an attempt to escape from local optima. In case of successful random jump, deterministic perturbation is performed again.

In [15] (PAIR), the *pairs of points* are used to define a projection for splitting the input space. For each pair of points, a normal vector defining a direction is constructed. The rest of training points is projected onto this vector and the projected values are utilized as thresholds defining splitting hyperplanes orthogonal to constructed normal vector. To reduce complexity, only the threshold halfway between the defining pair can be considered.

A nonparametric function estimation method called SUPPORT [6] is based on the analysis of *residuals* after regression to find a split. At the beginning, polynomial regression is performed on the training data. The points under the curve (negative residuals) present the first class, and the rest of points (positive or zero residuals) presents the second class. Afterwards, distribution analysis is applied to find a split.

4 Experimental evaluation

In this section, we compare the performances of the DTS-CMA-ES using the RFs as a surrogate model in several different settings to the original DTS-CMA-ES version, the original CMA-ES, and the lmm-CMA-ES on the noiseless part of the COCO platform [13, 14].

4.1 Experimental setup

The considered algorithms were compared on 24 noiseless single-objective continuous benchmark functions from the COCO testbed [13, 14] in dimensions $D = 2, 3, 5$, and 10 on 15 different instances per function. Each algorithm had a budget of $250D$ function evaluations to reach the target distance $\Delta f_T = 10^{-8}$ from the function optimum. The parameter settings of the tested algorithms are summarized in the following paragraphs.

The original CMA-ES was employed in its IPOP-CMA-ES version (Matlab code v. 3.61) with the following settings: the number of restarts = 4, IncPopSize = 2, $\sigma_{\text{start}} = \frac{8}{3}$, $\lambda = 4 + \lfloor 3 \log D \rfloor$. The remaining settings were left default.

The lmm-CMA-ES was utilized in its improved version published in [1]. The results have been downloaded from the COCO results data archive[1] in its GECCO 2013 settings.

The original DTS-CMA-ES was tested using the overall best settings from [2]: the probability of improvement as the uncertainty criterion, the population size $\lambda = 8 + \lfloor 6 \log D \rfloor$, and the number of originally-evaluated points $n_{\text{orig}} = \lceil 0.05\lambda \rceil$.

Considering decision tree settings, the five splitting methods from the following algoritms were employed: CART [4], SECRET [9], OC1 [17], SUPPORT [6], and PAIR [15]. Due to the different properties of individual splitting methods, the number of $\mathcal{L}_{\text{split}}$ evaluations was limited to $10D$ per node to restrict the algorithms which test a great number of hyperplanes. For the same reason, the number of tresholds generated by a projection of points to a hyperplane was set to 10 quantile-based values in CART, OC1, and to a median value in PAIR, and the searching an initial axis-aligned hyperplane in OC1 was limited to $\lceil \frac{10D}{3} \rceil$ $\mathcal{L}_{\text{split}}$ evaluations.

The RFs as a surrogate model were tested using the gradient boosting ensemble method. The maximum tree depth was set to 8, in accordance with [7]. In addition, the number of trees n_{tree}, the number of points N_t bootstrapped out of N archive points, and the number of randomly subsampled dimensions used for training the individual tree n_D were sampled from the values in Table 1.

The DTS-CMA-ES in combination with RFs was tested with the following settings: the probability of improvement as the uncertainty criterion, the population size $\lambda = 8 + \lfloor 6 \log D \rfloor$, and the number of originally-evaluated points n_{orig} with 4 different values $\lceil 0.05\lambda \rceil$, $\lceil 0.1\lambda \rceil$, $\lceil 0.2\lambda \rceil$, and $\lceil 0.4\lambda \rceil$. The rest of DTS-CMA-ES parameters have been taken identical to the overall best settings from [2].

4.2 Results

Result from experiments are presented in Figures 1–4 and also in Table 2. The graphs in Figures 1–4 depict the scaled best-achieved logarithms $\Delta_f^{\log}$ of median distances Δ_f^{med} to the functions optimum for the respective number of function evaluations per dimension (FE/D). Medians Δ_f^{med} (and in Figure 1 also 1st and 3rd quartiles) are calculated from 15 independent instances for each respective algorithm, function, and dimension. The scaled logarithms of Δ_f^{med} are calculated as

[1] `http://coco.gforge.inria.fr/data-archive/2013/lmm-CMA-ES_auger_noiseless.tgz`

Table 1: Experimental settings of RF: n_{orig} – number of originally-evaluated points , n_{tree} – number of trees in RF, N_t, n_D – number of tree points and dimensions. Split methods and n_{orig} are selected using full-factorial design, n_{tree}, N_t, and n_D are sampled.

parameter	values
n_{orig}	$\{\lceil 0.05\lambda \rceil, \lceil 0.1\lambda \rceil, \lceil 0.2\lambda \rceil, \lceil 0.4\lambda \rceil\}$
split	{CART, SECRET, OC1, SUPPORT, PAIR}
n_{tree}	$\{64, 128, 256, 512, 1024\}$
N_t	$\lceil \{0.25, 0.5, 0.75, 1\} \cdot N \rceil$
n_D	$\lceil \{0.25, 0.5, 0.75, 1\} \cdot D \rceil$

$$\Delta_f^{\log} = \frac{\log \Delta_f^{\text{med}} - \Delta_f^{\text{MIN}}}{\Delta_f^{\text{MAX}} - \Delta_f^{\text{MIN}}} \log_{10}\left(1/10^{-8}\right) + \log_{10} 10^{-8}$$

where Δ_f^{MIN} (Δ_f^{MAX}) is the minimum (maximum) $\log \Delta_f^{\text{med}}$ found among all the compared algorithms for the particular function f and dimension D between 0 and 250 FE/D. Such scaling enables the aggregation of $\Delta_f^{\log}$ graphs across arbitrary number of functions and dimensions (see Figures 2, 3, and 4). The values are scaled to the $[-8, 0]$ interval, where -8 corresponds to the minimal and 0 to the maximal distance. This visualization was chosen due to better ability to distinguish the differences in the convergence of tested algorithms in comparison with the default visualization used by the COCO platform.

We compare the statistical significance of differences in algorithms' performance on 24 COCO functions in $5D$ for separately two evaluation budgets utilizing the Iman and Davenport's improvement of the Friedman test [8]. Let $\#\text{FE}_\text{T}$ be the smallest number of FE on which at least one algorithm reached the target distance, i. e., satisfied $\Delta_f^{\text{med}} \leq \Delta f_T$, or $\#\text{FE}_\text{T} = 250D$ if no algorithm reached the target within $250D$ evaluations. The algorithms are ranked on each function with respect to Δ_f^{med} at a given budget of FE. The null hypothesis of equal performance of all algorithms is rejected for the higher function evaluation budget $\#\text{FEs} = \#\text{FE}_\text{T}$ ($p < 10^{-3}$), as well as for the lower budget $\#\text{FEs} = \frac{\#\text{FE}_\text{T}}{3}$ ($p < 10^{-3}$).

We test pairwise differences in performance utilizing the post-hoc test to the Friedman test [11] with the Bergmann-Hommel correction controlling the family-wise error. The numbers of functions at which one algorithm achieved a higher rank than the other are enlisted in Table 2. The table also contains the pairwise statistical significances.

The graphs in Figures 2 and 3 summarize the performance of five different split algorithms and four n_{orig} values from twenty different settings respectively. We found that the convergence of DTS-CMA-ES is quite similar regardless the split algorithm with slightly better results of SECRET and SUPPORT – the algorithms utilizing classification methods to find the splitting hyperplane between previously created clusters of training points. The results also show that lower n_{orig} values provide better performance in the initial phase of the optimization run and higher values are more successful starting from the 100-150 FE/D. Due to the presented results, the following comparisons contain the performances of the DTS-CMA-ES with $n_{\text{orig}} = \lceil 0.4\lambda \rceil$ in combination with RF using SECRET and SUPPORT as split algorithms.

As can be seen in Figures 1 and 4, the performance of RFs is considerably worse than the performance of GPs in combination with the DTS-CMA-ES and better than the performance of the original CMA-ES. RF model provides faster convergence from approximitely 100 FE/D on the regularly multimodal Rastrigin functions (f_3, f_4, and f_{15}) where the RF apparently does not prevent the original CMA-ES from exploiting the global structure of a function. The performance of RF-DTS-CMA-ES is noticably lower especially on the elipsoid (f_1, f_2, f_7, and f_{10}), Rastrigin (f_8, f_9), and ill-condition functions (f_{11-14}), where smooth models are much more convenient for regression. On the other hand, RFs help the CMA-ES to convergence especially on the multimodal functions f_{16-19}, where the performance of RF-DTS-CMA-ES is the best of all compared algorithms.

5 Conclusions & Future work

In this paper, we have compared the RF model using gradient boosting as the ensemble method with the GP regression model, both used as surrogate models in the DTS-CMA-ES algorithm. Different methods of space splitting in regression trees were investigated.

The split algorithms SECRET and SUPPORT based on the classification of the input points provide slightly better performance as to the CMA-ES convergence than the other algorithms tested. Moreover, the performance of DTS-CMA-ES using RFs differs according to the number of originally-evaluated points: the lower their number, the sooner the algorithm converges, possibly to a local optimum, which makes convergence to the global one more difficult. We found that the RF model usually reduces the number of fitness evaluations required by the CMA-ES, especially on multi-modal functions, where the provided speed-up was the best among all compared algorithms for a number of evaluations higher than approximitely 110 FE/D.

A possible perspective for future research is to improve RF models by implementing non-constant (linear, quadratic) models to regression tree leaves, which could make the RFs prediction more convenient for smooth functions. Investigation of other split algorithms could also bring interesting results. Another perspective for future research is an automatical selection of the most convenient surrogate model for the CMA-ES inside the algorithm itself.

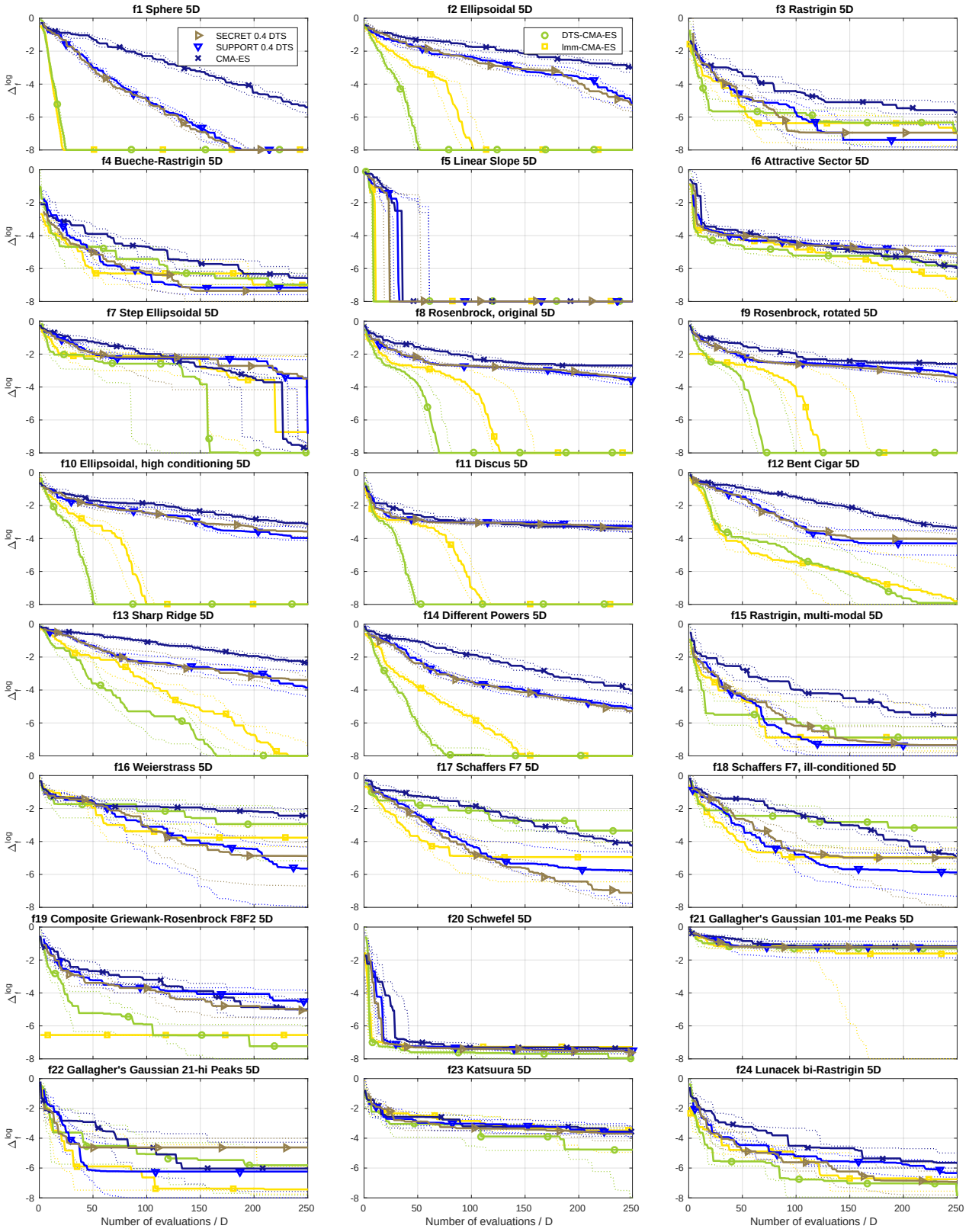


Figure 1: Medians (solid) and 1st/3rd quartiles (dotted) of the distances to the optima of 24 COCO benchmarks in 5D for algorithms CMA-ES, DTS-CMA-ES, lmm-CMA-ES, and 2 RF settings of DTS-CMA-ES. Medians/quartiles were calculated across 15 independent instances for each algorithm and are shown in the $\log_{10}$ scale.

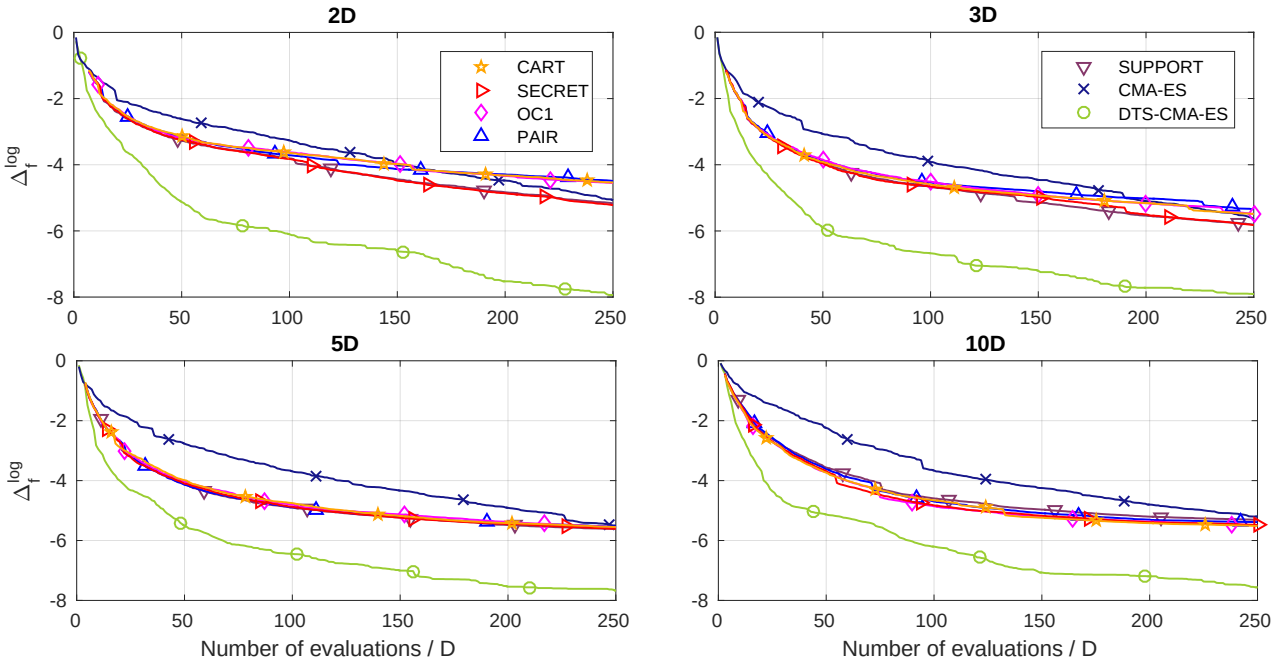


Figure 2: Scaled median distances $\Delta_f^{\log}$ of decision tree split algorithms averaged over all 24 COCO functions in 2D, 3D, 5D, and 10D for algorithms CART, SECRET, OC1, PAIR, and SUPPORT in combination with the DTS-CMA-ES and all tested numbers of originally-evaluated points.

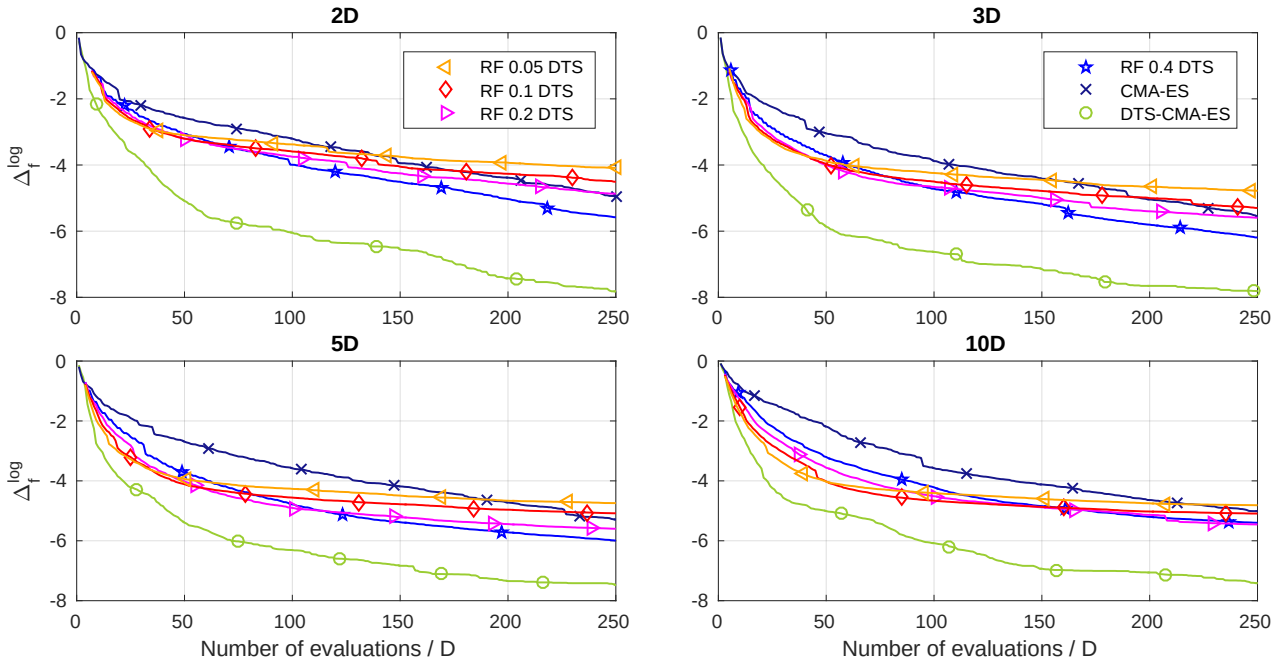


Figure 3: Scaled median distances $\Delta_f^{\log}$ of the DTS-CMA-ES with RFs comparing different numbers of originally-evaluated points averaged over all 24 COCO functions in 2D, 3D, 5D, and 10D for values $\lceil 0.05\lambda \rceil$, $\lceil 0.1\lambda \rceil$, $\lceil 0.2\lambda \rceil$, and $\lceil 0.4\lambda \rceil$ summarized accross all tested splitting algorithms.

Table 2: A pairwise comparison of the algorithms in 5*D* over the COCO for different evaluation budgets. The number of wins of *i*-th algorithm against *j*-th algorithm over all benchmark functions is given in *i*-th row and *j*-th column. The asterisk marks the row algorithm being significantly better than the column algorithm according to the Friedman post-hoc test with the Bergmann-Hommel correction at family-wise significance level $\alpha = 0.05$.

5D	SECRET 0.4 DTS		SUPPORT 0.4 DTS		CMA-ES		DTS-CMA-ES		lmm-CMA-ES	
$^{\#FEs}/_{\#FE_T}$	1/3	1	1/3	1	1/3	1	1/3	1	1/3	1
SECRET 0.4 DTS	—	—	11.5	11	23*	21*	8	6	5	8
SUPPORT 0.4 DTS	12.5	13	—	—	24*	21*	7	7	7	8
CMA-ES	1	3	0	3	—	—	3	4	1	3
DTS-CMA-ES	16	18	17	17	21*	20*	—	—	14	14
lmm-CMA-ES	19	16	17	16	23*	21*	10	10	—	—

Acknowledgements

The reported research was supported by the Czech Science Foundation grant No. 17-01251, by the Grant Agency of the Czech Technical University in Prague with its grant No. SGS17/193/OHK4/3T/14, and by Specific College Research project number 260 453. Further, access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Research, Development, and Innovations Infrastructures" (CESNET LM2015042), is greatly appreciated.

References

[1] A. Auger, D. Brockhoff, and N. Hansen. Benchmarking the local metamodel CMA-ES on the noiseless BBOB'2013 test bed. In *Genetic and Evolutionary Computation Conference, GECCO '13, Amsterdam, The Netherlands, July 6-10, 2013, Companion Material Proceedings*, pages 1225–1232, 2013.

[2] L. Bajer. *Model-based evolutionary optimization methods*. PhD thesis, Faculty of Mathematics and Physics, Charles University in Prague, Prague, 2018.

[3] L. Bajer, Z. Pitra, and M. Holeňa. Benchmarking Gaussian processes and random forests surrogate models on the BBOB noiseless testbed. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO Companion '15, pages 1143–1150, New York, NY, USA, July 2015. ACM.

[4] L. Breiman. *Classification and regression trees*. Chapman & Hall/CRC, 1984.

[5] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[6] P. Chaudhuri, M.-C. Huang, W.-Y. Loh, and R. Yao. Piecewise-polynomial regression trees. *Statistica Sinica*, 4(1):143–167, 1994.

[7] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. KDD '16, pages 785–794. ACM, 2016.

[8] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[9] A. Dobra and J. Gehrke. SECRET: A scalable linear regression tree algorithm. KDD '02, pages 481–487. ACM, 2002.

[10] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[11] S. García and F. Herrera. An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694, 2008.

[12] N. Hansen. The CMA Evolution Strategy: A Comparing Review. In *Towards a New Evolutionary Computation*, number 192, pages 75–102. Springer, 2006.

[13] N. Hansen, A. Auger, S. Finck, and R. Ros. Real-parameter black-box optimization benchmarking 2012: Experimental setup. Technical report, INRIA, 2012.

[14] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Technical Report RR-6829, INRIA, 2009. Updated February 2010.

[15] G. E. Hinton and M. Revow. Using pairs of data-points to define splits for decision trees. In *Advances in Neural Information Processing Systems*, volume 8, pages 507–513. MIT Press, 1996.

[16] S. Kern, N. Hansen, and P. Koumoutsakos. Local Metamodels for Optimization Using Evolution Strategies. In *Parallel Problem Solving from Nature - PPSN IX*, volume 4193 of *Lecture Notes in Computer Science*, pages 939–948. Springer Berlin Heidelberg, 2006.

[17] S. K. Murthy, S. Kasif, and S. Salzberg. A system for induction of oblique decision trees. *J. Artif. Int. Res.*, 2(1):1–32, 1994.

[18] Z. Pitra, L. Bajer, and M. Holeňa. Doubly trained evolution control for the Surrogate CMA-ES. In *PPSN XIV Proceedings*, pages 59–68. Springer, 2016.

[19] Z. Pitra, L. Bajer, J. Repický, and M. Holeňa. Ordinal ver-

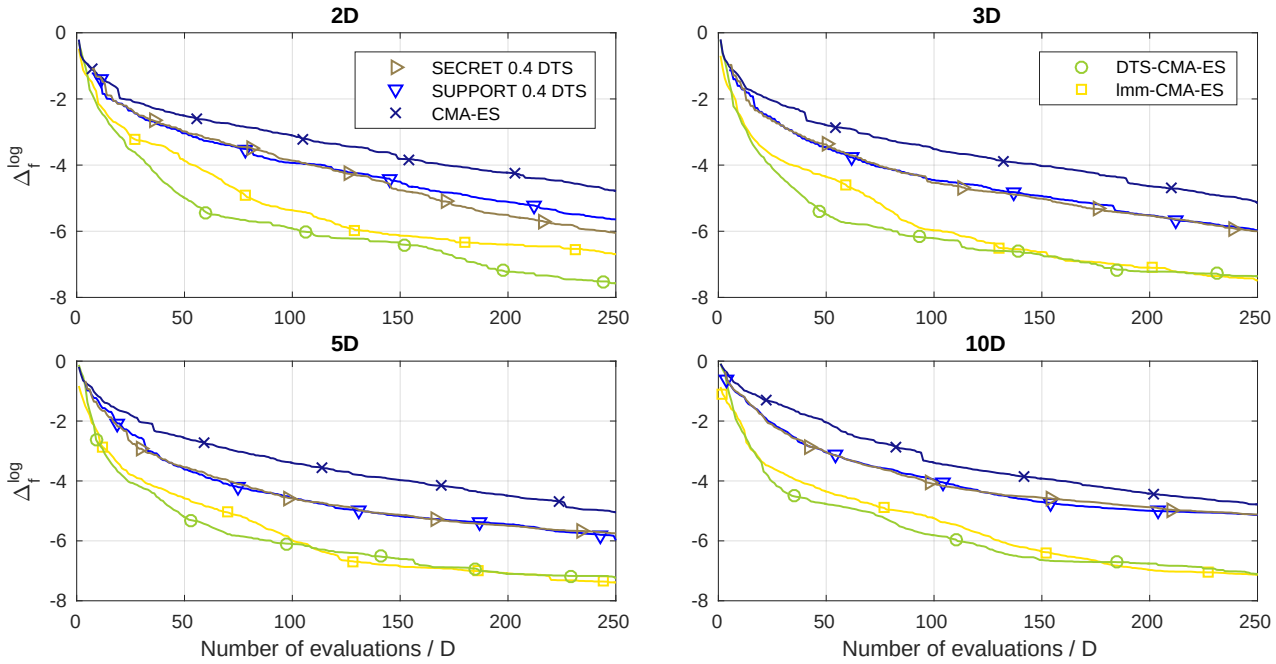


Figure 4: Scaled median distances $\Delta_f^{\log}$ averaged over all 24 COCO functions in $2D$, $3D$, $5D$, and $10D$ for algorithms CMA-ES, DTS-CMA-ES, lmm-CMA-ES, and 2 RF settings of DTS-CMA-ES.

sus metric Gaussian process regression in surrogate modelling for CMA evolution strategy. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '17, pages 177–178, New York, NY, USA, 2017. ACM.

[20] Z. Pitra, L. Bajer, J. Repický, and M. Holeňa. Overview of surrogate-model versions of covariance matrix adaptation evolution strategy. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '17, pages 1622–1629, New York, NY, USA, 2017. ACM.

[21] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptative computation and machine learning series. MIT Press, 2006.

S. Krajči (ed.): ITAT 2018 Proceedings, pp. 80–85
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Petra Vidnerová and Roman Neruda

Asynchronous Evolution of Convolutional Networks

Petra Vidnerová, Roman Neruda

Czech Academy of Sciences
Institute of Computer Science
petra@cs.cas.cz

Abstract: Due to many successful practical applications, deep neural networks and convolutional networks have become the state-of-art machine learning methods recently. The choice of network architecture for the task at hand is typically made by trial and error.

This work deals with an automatic data-dependent architecture design. We propose an algorithm for optimization of architecture of convolutional network based on asynchronous evolution. The algorithm is inspired by and designed directly for the Keras library which is one of the most common implementations of deep neural networks. The proposed algorithm is successfully tested on MNIST and Fashion-MNIST data sets.

1 Introduction

Various architectures of deep neural networks (DNN) have become the state-of-art methods in many fields of machine learning in recent years. They have been applied to various problems, including image recognition, speech recognition, and natural language processing [7, 11].

Deep neural networks are feed-forward neural networks with multiple hidden layers between the input and output layer. The layers typically have different units depending on the task at hand. Among the units, there are traditional perceptrons, where each unit (neuron) realizes a nonlinear function, such as the *sigmoid* function, or the rectified linear unit (*ReLU*).

Convolutional networks (CNN) are family of DNN. They typically have three types of layers – convolutional layers, max-pooling layers, and dense (i.e. fully-connected) layers. For the most common case of image processing, convolutional layers perform convolution of an input image to abstract high-level features. They are defined by a set of learnable filters. Max-pooling layers reduce the size of representation, their function is fixed and they are not learnable. Dense layers are usually used as the last layers of the network to perform the actual classification.

While the learning of weights (including filters) of the CNN is done by algorithms based on the stochastic gradient descent, the choice of architecture, including a number and sizes of layers, number and size of convolutional filters, size of pools in pooling-layers, and a type of activation function, is done manually by the user. However, the choice of architecture has an important impact on the performance of the CNN. Some kind of expertise is needed, and usually a trial and error method is used in practice.

In this work we exploit a fully automatic design of CNNs. We investigate the use of evolutionary algorithms for evolution of a CNN architecture. There are not many studies on evolution of CNN since such approach has very high computational requirements. To keep the search space as small as possible, we simplify our model focusing on implementation of CNN in the Keras library [3] that is a widely used tool for practical applications of DNNs and CNNs.

The approach described in this paper extends our previous results for evolving DNNs limited to networks with dense layers only [25, 24]. The proposed algorithm is evaluated on the MNIST and Fashion-MNIST data sets that are both classification tasks of small gray-scale images.

The paper is organized as follows. Next Section reviews related work. Section 3 describes the main ideas of our approach. Section 4 explains the main ideas of asynchronous evolution. Section 5 summarises the results of our experiments, and finally Section 6 brings conclusion.

2 Related Work

Neuroevolution represents an attempt to train a neural network by means of evolutionary techniques [5]. In traditional neuroevolution, no gradient descent is usually involved, and both architecture and weights of the network undergo the evolutionary process. However, because of large computational requirements the applications are limited to small networks.

There were quite many attempts on architecture optimization via evolutionary process (e.g. [22, 1]) in previous decades. Successful evolutionary techniques evolving the structure of feed-forward and recurrent neural networks include NEAT [20], HyperNEAT [19] and CoSyNE [6] algorithms.

On the other hand, studies dealing with evolution of deep neural networks and convolutional networks started to emerge only very recently. The training of one DNN usually requires hours or days of computing time, quite often utilizing GPU processors for speedup. Naturally, the evolutionary techniques requiring thousands of training trials were not considered a feasible choice. Nevertheless, there are several approaches to reduce the overall complexity of neuroevolution for DNN. Still due to limited computational resources, the studies usually focus only on parts of network design.

For example, in [14] CMA-ES is used to optimize hyperparameters of DNNs. In [10] the unsupervised convolutional networks for vision-based reinforcement learning are studied, the structure of CNN is held fixed, and only a small recurrent controller is evolved. However, the recent paper [17] presents a simple distributed evolutionary strategy that is used to train relatively large recurrent network with competitive results on reinforcement learning tasks.

In [16] automated method for optimizing deep learning architectures through evolution is proposed, extending existing neuroevolution methods. Authors of [4] sketch a genetic approach for evolving a deep autoencoder network enhancing the sparsity of the synapses by means of special operators. The paper [15] presents two versions of an evolutionary and co-evolutionary algorithm for design of DNN with various transfer functions. Finally, [21] proposes genetic programming to evolve CNNs.

3 Our Approach

In our approach we use asynchronous evolution to search for optimal architecture of CNN, while the weights are learned by gradient based technique.

The main idea of our approach is to keep the search space as small as possible, therefore the architecture specification is simplified. It directly follows the implementation of CNN in Keras library, where networks are defined layer by layer. Layer is specified by its type – convolutional, max-pooling, dense. Dense layers are defined by a number of neurons, type of an activation function (all neurons in one layer have the same type of their activation function), and the type of regularization (such as dropout). Convolutional layers are defined by number of filters, size of the filter, and possibly the type of an activation function and type of regularization. Max-pooling layers are defined by the size of pool.

In this paper, we limit to networks that can be split into two parts.The first part is a preprocessing part, it contains only convolutional and max-pooling layers, and it is responsible for the preprocessing of the input and abstracting high-level features. The second part is a classifier and it consists of dense layers. Such architecture corresponds to the original proposal of the LeNet architecture [12] (Fig. 1).

3.1 Individuals

In order to apply genetic algorithm (GA) to the search for an optimal CNN architecture, we have to be able to encode the architecture by an individual of the GA.

Our proposal of encoding closely follows the CNN description and implementation in the Keras [3] model *Sequential*. The model implemented as *Sequential* is built layer by layer, similarly the GA individual consists of blocks representing individual layers.

$$
\begin{aligned}
&I = (I_1, I_2),\\
&I_1 = ([type, params]_1, \ldots, [type, params]_{H1})\\
&I_2 = ([size, dropout, act]_1, \ldots, [size, dropout, act]_{H2})
\end{aligned}
$$

where I_1 and I_2 are the convolutional and dense part, respectively, $H1, H2$ is the number of layers in convolutional and dense part, respectively. The blocks in convolutional part encode $type \in \{\texttt{convolutional}, \texttt{max}-\texttt{pooling}\}$ type of layer and *params* other parameters of the layer (for convolutional layer it is number of filters, size of filter, and activation function; for max-pooling layer it is only size of pool). The blocks in dense part code dense layers, so they consist of *size* the number of neurons, $drop$ the dropout rate (zero value represents no dropout), $act \in \{\texttt{relu}, \texttt{tanh}, \texttt{sigmoid}, \texttt{hardsigmoid}, \texttt{linear}\}$ activation function.

3.2 Genetic Operators

To produce new individuals in genetic algorithm we use recombination operators *crossover* and *mutation*.

Crossover The *crossover* operator combines two parent individuals and produces two offspring individuals. It is implemented as one-point crossover, where the crossing point is determined at random, but on the border of a block only. The two parts of the individual are crossed over separately, so if parents are $I = (I_1, I_2)$ and $J = (J_1, J_2)$ we run $crossover(I_1, J_1)$ and $crossover(I_2, J_2)$.

Let the two parents be:

$$I_{p1} = (B_1^{p1}, B_2^{p1}, \ldots, B_k^{p1})$$

$$I_{p2} = (B_1^{p2}, B_2^{p2}, \ldots, B_l^{p2}),$$

then, the crossover produces offspring:

$$I_{o1} = (B_1^{p1}, \ldots, B_{cp1}^{p1}, B_{cp2+1}^{p2}, \ldots, B_l^{p2})$$

$$I_{o1} = (B_1^{p2}, \ldots, B_{cp2}^{p2}, B_{cp1+1}^{p1}, \ldots, B_k^{p1}),$$

where $cp_1 \in \{1, \ldots, k-1\}$ and $cp_2 \in \{1, \ldots, l-1\}$.

Thus, only the whole layers are interchanged between individuals.

Mutation The *mutation* operator brings random changes to an individual. Each time an individual is mutated, one of the following mutation operators is randomly chosen (each of mutation operators has its own probability):

- mutateLayer - introduces random changes to one randomly selected layer.
- addLayer - one randomly generated block is inserted at random position. If it is inserted to the first part of the individual, its either convolutional layer or max-pooling layer; otherwise it is dense layer.

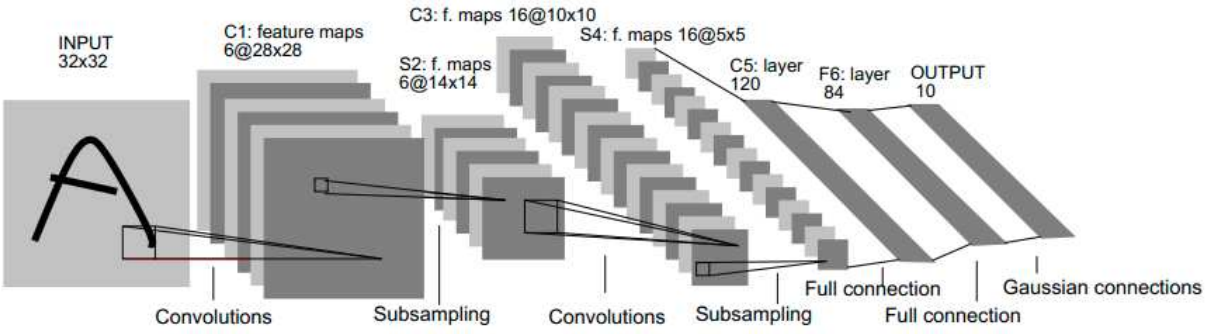


Figure 1: Convolutional neural network [12].

- delLayer - one randomly selected block is deleted.

When *mutateLayer* is performed, again one of the available operators is chosen. For dense layers they are:

- changeLayerSize - the number of neurons is changed. The Gaussian mutation is used, the final number is rounded (since size has to be an integer).
- changeDropOut - the dropout rate is changed using the Gaussian mutation.
- changeActivation - the activation function is changed, randomly chosen from the list of available activations.

For max-pooling layers:

- changePoolSize - the size of pooling is changed.

For convolutional layers:

- changeNumberOfFilters - the number of filters is changed. The Gaussian mutation is used, the final number is rounded.
- changeFilterSize - the size of the filter is changed.
- changeActivation - the activation function is changed, randomly chosen from the list of available activations.

3.3 Fitness

Fitness function should reflect a quality of the network represented by an individual. To assess the generalization ability of the network represented by the individual we use a crossvalidation error. The lower the crossvalidation error, the higher the fitness of the individual.

Classical k-fold crossvalidation is used, i.e. the training set is split into k-folds and each time one fold is used for testing and the rest for training. The mean loss function on the testing set over k run is evaluated.

For the classification tasks, categorical crossentropy is used as the loss function.

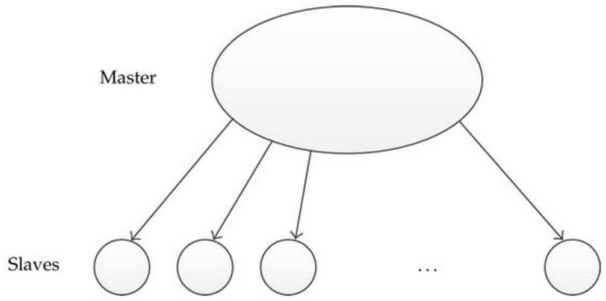


Figure 2: Master-slave parallel computational model.

3.4 Selection

As a parental selection operator, the tournament selection is used in our algorithm. It works as follows, in each turn of the tournament, k individuals are selected at random, and the one with the highest fitness—in our case the one with the lowest crossvalidation error—is selected.

4 Asynchronous Evolution

In classical genetic algorithm the individuals are evaluated in generations. In each generation, new individuals are produced based on operators selection, mutation, and crossover, their fitness is evaluated and they replace the old generation. The fitness evaluations are independent and can be done in parallel.

There are many approaches to parallelization of genetic algorithms. In general, there are three classes of parallel GA approaches – single population master-slave model, single population fine-grained, and multi-population coarse grained model (see [2, 9] for more details).

In our work we use the master-slave parallelization method (Fig. 2). It works with single population of individuals, and the evaluations of individuals are performed in parallel – the master stores the population and the slaves

evaluate the fitness. The algorithm itself is same as for serial GA, each individual may compete and mate with any other.

The fitness evaluation is parallelized so that a fraction of the population is assigned to each of the processors. Communication occurs only when a slave receives an individual to evaluate and when a slave returns a fitness value.

The algorithm is normally defined as synchronous, i.e. the master waits for the slaves to receive the fitness values for all the population and only then it proceeds to the next generation. Such synchronous master-slave algorithm has exactly the same properties as a simple sequential algorithm.

However, the disadvantage of such synchronous parallel approach is that in general, not all fitness evaluations require same time. In our particular case, some individuals represent large networks and require long time to evaluate, on the other hand there might be very small networks that are evaluated much faster. At the end of each generation, there are processors that are already finished and waiting for the ones evaluating large networks.

As a solution to this problem we have chosen to use *asynchronous evolution* [18]. In asynchronous evolution there is no notion of generations, but as soon as there is a free processor, a new individual is generated and send for evaluation. Such approach may significantly improve the usage of computationally resources. The scheme of the algorithm is presented in Alg. 1.

One of the features of asynchronous approach is that it is naturally biased towards solution with faster fitness evaluation. In our case we consider this feature to be an advantage, because we are more interested in smaller networks with shorter learning time. As our practical experiments in the next section imply, this feature is not harmless, it seems in fact not sufficient and maybe a further discrimination of larger networks in fitness function can be beneficial.

The (synchronous and asynchronous) master-slave approach can be easily and efficiently implemented both on shared-memory and distributed-memory parallel computers. On a shared-memory multiprocessor, the population is stored in memory and each slave process can access the individuals assigned to itself. On a distributed-memory computer, the master process is responsible for storing the population, sending the individuals to other processes (slaves), collecting the results and producing new generation by genetic operators.

Algorithm 1 Asynchronous EA

```
procedure ASYNCEA(MINPOPSIZE,POPSIZE)
    P ← ∅
    while |P| < minPopSize do
        if not node available then
            wait()
        end if
        while node available do
            ind ← RandomIndividual()
            evaluate(ind)
        end while
        evaluatedInd ← getEvaluatedIndividual()
        P ← P ∪ {evaluatedInd}
    end while
    ind ← produceIndividual()
    evaluate(ind)
    while termination criterion not met do
        evaluatedInd ← getEvaluatedIndividual()
        P ← P ∪ {evaluatedInd}
        if |P| > popSize then
            discard the worst individual from P
        end if
        ind ← produceIndividual()
        evaluate(ind)
    end while
end procedure
```

5 Experiments

For our experiments we have chosen the well known MNIST data set [13] and Fashion-MNIST data set [26].

Each data set contains 70 000 images of 28×28 pixel. 60 000 are used for training, 10 000 for testing. In MNIST, there are images of handwritten digits (see Fig. 3), while in Fashion-MNIST are images of fashion objects (Fig. 4).

Our implementation of the proposed algorithm is available at [23].

We have run the algorithm with population of 30 individuals on 10 processors for one week.

When the best individual is obtained, the corresponding network is built and trained on the whole training set and evaluated on the test set, which was also done for the baseline model designed by human. For both models, the RMSProp algorithm for 20 epochs was used.

The resulting classification accuracy on the test set is listed in Tab. 1 and Tab. 4 for MNIST and Fashion-MNIST data sets respectively. The obtained accuracies can be further improved (especially in case of Fashion-MNIST) by tuning the parameters of RMSProp (default setup was used in our case). On both datasets the obtained evolved network gives competitive results to the baseline model.

In Tab. 2 and Tab. 3, there are listings of baseline and evolved architectures for MNIST and Fashion-MNIST, respectively. The `conv #32` stands for convolutinal layer with 32 filters, `dense #128` stands for dense layer with 128 neurons, and `pool` stands for max-pooling layer, etc.

On the MNIST data set, the evolved architecture has even less number of weights than the baseline model. However, on the Fashion-MNIST the evolution was not so successful and the evolved architecture is quite bloated. The architecture complexity is not reflected in the fitness function directly, but the asynchronous evolution should tend to prefer smaller networks.

We have also compared the synchronous parallelization with asynchronous one in terms of time requirements. We

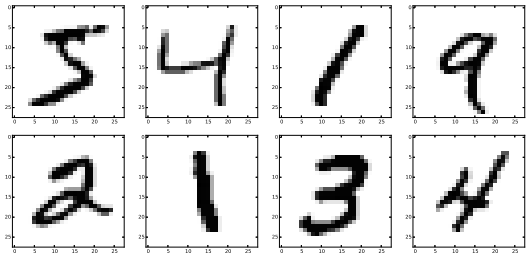


Figure 3: Example of MNIST data set samples.

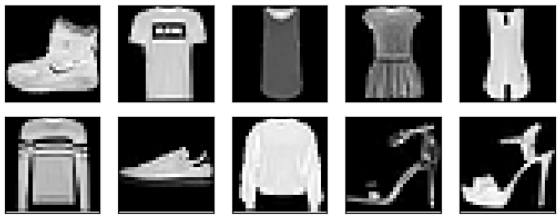


Figure 4: Example of Fashion-MNIST data set samples.

model	avg	std	min	max
baseline	98.97	0.07	98.84	99.13
evolved	**99.25**	0.09	99.10	99.37

Table 1: Test accuracies on the MNIST dataset.

Baseline network
conv #32 kernelsize=3 activation=relu
conv #32 kernelsize=3 activation=relu
pool poolsize=2
dense #128 dropout=0.5 activation=relu
Trainable params: 600,810

Evolved network
conv #22 kernelsize=2 activation=tanh
conv #31 kernelsize=5 activation=linear
pool poolsize=3
conv #33 kernelsize=5 activation=relu
dense #143 dropout=0.4 activation=relu
dense #42 dropout=0.0 activation=tanh
Trainable params: 431,659

Table 2: Baseline and evolved network for MNIST.

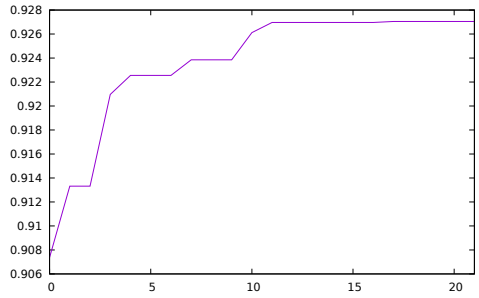


Figure 5: The fitness function through evolution.

Baseline network
conv #32 kernelsize=3 activation=leakyRelu
pool poolsize=2
conv #64 kernelsize=3 activation=leakyRelu
pool poolsize=2
conv #128 kernelsize=3 activation=leakyRelu
pool poolsize=2
dense #128 dropout=0.3 activation=leakyRelu
Trainable params: 356,234

Evolved network
conv #46 kernelsize=3 activation=relu
conv #15 kernelsize=3 activation=relu
conv #36 kernelsize=4 activation=relu
conv #13 kernelsize=3 activation=relu
conv #36 kernelsize=3 activation=relu
pool poolsize=2
dense #235 dropout=0.4 activation=hard_sigmoid
dense #130 dropout=0.3 activation=tanh
Trainable params: 1,714,219

Table 3: Baseline and evolved network for Fashion-MNIST.

model	avg	std	min	max
baseline	91.64	0.37	90.77	91.97
evolved	**92.32**	0.52	91.07	92.86

Table 4: Test accuracies on the Fashion-MNIST dataset.

have run both algorithms on 5 processors for 4 days with population size 20. The asynchronous version made 140 fitness evaluations, while the synchronous version 100 fitness evaluations. So the asynchronous version may bring quite important time saving.

6 Conclusion

We have proposed an algorithm for automatic design of convolutional networks based on asynchronous evolution. The algorithm was tested in experiments on two image classification tasks, the MNIST and Fashion-MNIST data sets. The evolved networks were compared to baseline models, and they achieved competitive results (in terms of slightly better classification accuracies). We have shown that it is possible to automatically find solutions comparable to those designed by human expert.

The main limitation of the presented algorithm is its time complexity. The possibility to trade human expert knowledge for computational resources should be seen as an advantage in several scenarios. Our main motivation is to develop an autonomous system capable of creating machine learning models without human intervention. This may be useful for the cases where no expert is available or a new task without prior experience is encountered. Also, in critical cases where even a small performance gain is necessary, our approach has demonstrated its usability.

One direction of our future work is to try to lower the number of fitness evaluations using surrogate modelling [8] or to investigate other types of parallel evolutionary algorithms (such as multi-deme GA [9]). We also plan to tune automatically the learning algorithm, i.e. search for other hyper-parameters, such as the type of learning algorithm, learning rates, etc.

Acknowledgment

This work was partially supported by the Czech Grant Agency grant 18-23827S and institutional support of the Institute of Computer Science RVO 67985807.

Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum provided under the programme "Projects of Large Research, Development, and Innovations Infrastructures" (CESNET LM2015042), is greatly appreciated.

References

[1] J. Arifovic and R. Gençay. Using genetic algorithms to select architecture of a feedforward artificial neural network. *Physica A: Statistical Mechanics and its Applications*, 289(3–4):574 – 594, 2001.

[2] E. Cantú-Paz. A survey of parallel genetic algorithms. *CALCULATEURS PARALLELES*, 10, 1998.

[3] F. Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[4] O. E. David and I. Greental. Genetic algorithms for evolving deep neural networks. In *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation*, GECCO Comp '14, pages 1451–1452, New York, NY, USA, 2014. ACM.

[5] D. Floreano, P. Dürr, and C. Mattiussi. Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1(1):47–62, 2008.

[6] F. Gomez, J. Schmidhuber, and R. Miikkulainen. Accelerated neural evolution through cooperatively coevolved synapses. *Journal of Machine Learning Research*, pages 937–965, 2008.

[7] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[8] Y. Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1(2):61 – 70, 2011.

[9] D. S. Knysh and Victor M. Kureichik. Parallel genetic algorithms: a survey and problem state of the art. *Journal of Computer and Systems Sciences International*, 49(4):579–589, Aug 2010.

[10] J. Koutník, J. Schmidhuber, and F. Gomez. Evolving deep unsupervised convolutional networks for vision-based reinforcement learning. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, GECCO '14, pages 541–548, New York, NY, USA, 2014. ACM.

[11] Y. Lecun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 5 2015.

[12] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[13] Y. LeCun and C. Cortes. The mnist database of handwritten digits, 2012.

[14] I. Loshchilov and F. Hutter. CMA-ES for hyperparameter optimization of deep neural networks. *CoRR*, abs/1604.07269, 2016.

[15] T. H. Maul, A. Bargiela, S.-Y. Chong, and A. S. Adamu. Towards evolutionary deep neural networks. In Flaminio Squazzoni, Fabio Baronio, Claudia Archetti, and Marco Castellani, editors, *ECMS 2014 Proceedings*. European Council for Modeling and Simulation, 2014.

[16] R. Miikkulainen, J. Zhi Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, and B. Hodjat. Evolving deep neural networks. *CoRR*, abs/1703.00548, 2017.

[17] T. Salimans, J. Ho, X. Chen, and I. Sutskever. Evolution Strategies as a Scalable Alternative to Reinforcement Learning. *ArXiv e-prints*, March 2017.

[18] E. O. Scott and K. A. De Jong. Understanding simple asynchronous evolutionary algorithms. In *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*, pages 85–98, 2015.

[19] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci. A hypercube-based encoding for evolving large-scale neural networks. *Artif. Life*, 15(2):185–212, April 2009.

[20] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.

[21] M. Suganuma, S. Shirakawa, and T. Nagao. A genetic programming approach to designing convolutional neural network architectures. *CoRR*, abs/1704.00764, 2017.

[22] B. u. Islam, Z. Baharudin, M. Q. Raza, and P. Nallagownden. Optimization of neural network architecture using genetic algorithm for load forecasting. In *2014 5th International Conference on Intelligent and Advanced Systems (ICIAS)*, pages 1–6, June 2014.

[23] P. Vidnerová. GAKeras. `github.com/PetraVidnerova/GAKeras`, 2017.

[24] P. Vidnerová and R. Neruda. Evolution strategies for deep neural network models design. In *Proceedings ITAT 2017: Information Technologies - Applications and Theory. Aachen & Charleston: Technical University & CreateSpace Independent Publishing Platform, 2017 - (Hlaváčová, J.), CEUR Workshop Proceedings, V-1885*, pages 159–166.

[25] P. Vidnerova and R. Neruda. Evolving keras architectures for sensor data analysis. In *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 109–112, Sept 2017.

[26] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

S. Krajči (ed.): ITAT 2018 Proceedings, pp. 86–91
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Vera Kůrková and Marcello Sanguineti

Probabilistic Bounds on Complexity of Networks Computing Binary Classification Tasks

Věra Kůrková[1] Marcello Sanguineti[2]

[1] Institute of Computer Science, Czech Academy of Sciences, Prague, Czech Republic
vera@cs.cas.cz,
WWW home page: http://www.cs.cas.cz/∼vera
[2] DIBRIS, University of Genoa, Genoa, Italy
marcello.sanguineti@unige.it
WWW home page: http://www.dist.unige.it/msanguineti/

Abstract: Complexity of feedforward networks computing binary classification tasks is investigated. To deal with unmanageably large number of these tasks on domains of even moderate sizes, a probabilistic model characterizing relevance of the classification tasks is introduced. Approximate measures of sparsity of networks computing randomly chosen functions are studied in terms of variational norms tailored to dictionaries of computational units. Probabilistic lower bounds on these norms are derived using the Chernoff-Hoeffding Bound on sums of independent random variables, which need not be identically distributed. Consequences of the probabilistic results on the choice of dictionaries of computational units are discussed.

1 Introduction

It has long been known that one-hidden-layer (shallow) networks with computational units of many common types can exactly compute any function on a finite domain [8]. In particular, they can perform any binary-classification task. Proofs of theorems on the universal approximation and representation properties of feedforward networks guarantee their power to express wide classes of functions, but do not deal with the efficiency of such representations. Typically, such arguments assume that the number of units is unbounded or is as large as the size of the domain of functions to be computed. For large domains, implementations of such networks might not be feasible.

A proper choice of a network architecture and a type of its units can, in some cases, considerably reduce network complexity. For example, a classification of points in the d-dimensional Boolean cube $\{0,1\}^d$ according to the parity of the numbers of 1's cannot be computed by a Gaussian SVM network with less than 2^{d-1} support vectors [3] (i.e., it cannot be computed by a shallow network with less than 2^{d-1} Gaussian SVM units). On the other hand, it is easy to show that the parity function (as well as any generalized parity, the set of which forms the Fourier basis) can be computed by a shallow network with merely $d+1$ Heaviside perceptrons [12].

The basic measure of sparsity of a network with a single linear output is the number of its nonzero output weights. The number of nonzero entries of a vector in $\mathbb{R}^n$ is called "l_0-pseudonorm". The quotation marks are used as l_0 is not homogenous and its "unit ball" is unbounded and nonconvex. Thus, minimization of the number of nonzero entries of an output-weight vector is a difficult nonconvex optimization task. Minimization of "l_0-pseudonorm" has been studied in signal processing, where it was shown that in some cases it is NP-hard [20].

A good approximation of convexification of "l_0-pseudonorm" is the l_1-norm [17]. In neurocomputing, l_1-norm has been used as a stabilizer in weight-decay regularization techniques [7]. In statistical learning theory, it l_1-norm plays an important role in LASSO regularization [19].

Networks with large l_1-norms of output-weight vectors have either large numbers of units or some of the weights are large. Both are not desirable: implementation of networks with large numbers of units might not be feasible and large output weights might lead to instability of computation. The minimum of the l_1-norms of output-weight vectors of all networks computing a given function is bounded from below by the variational norm tailored to a type of network units, which is a critical factor in estimates of upper bounds on network complexity [10, 11].

To identify and explain design of networks capable of efficient classifications, one has to focus on suitable classes of tasks. Even on a domain of a moderate size, there exists an enormous number of functions representing multi-class or binary classifications. For example, when the size of a domain is equal to 80, then the number of classifications into 10 classes is 10^{80} and when its size is 267, then the number of binary classification tasks is 2^{267}. These numbers are larger than the estimated number 10^{78} of atoms in the observable universe (see, e.g., [15]). Obviously, most classification tasks on such domains are not likely to be relevant for neurocomputing, as they do not model any task of practical interest.

In this paper, we investigate how to choose dictionaries of network units such that binary classification tasks can be efficiently solved. We assume that elements of a finite domain in $\mathbb{R}^d$ represent vectors of features, measurements, or observations for which some prior knowledge is available about probabilities that a presence of each of

these features implies the property described by one of the classes. For example, when vectors in the domain represent ordered sets of medical symptoms, certain values of some of these symptoms might indicate a high probability of some diagnosis, while others might indicate a low probability or be irrelevant.

For sets of classification tasks endowed with product probability distributions, we explore suitability of dictionaries of computational units in terms of values of variational norms tailored to the dictionaries. We analyze consequences of the concentration of measure phenomena which imply that with increasing sizes of function domains, correlations between network units and functions tend to concentrate around their mean or median values. We derive lower bounds on variational norms of functions to be computed and on l_1-norms of output-weight vectors of networks computing these functions. To obtain the lower bounds, we apply the Chernoff-Hoeffding Bound [5, Theorem 1.11] on sums of independent random variables not necessarily identically distributed. We show that when a priori knowledge of classification tasks is limited, then sparsity can only be achieved with large sizes of dictionaries. On the other hand, when such knowledge is biased, then there exist functions with which most functions on a large domain are highly correlated. If some of these functions is close to an element of a dictionary, then most functions can be well approximated by sparse networks with units from the dictionary.

The paper is organized as follows. In Section 2, we introduce basic concepts on feedforward networks, dictionaries of computational units, and approximate measures of network sparsity. In Section 3, we propose a probabilistic model of classification tasks and analyze properties of approximate measures of sparsity using the Chernoff-Hoeffding Bound. In Section 4, we derive estimates of probability distributions of values of variational norms and analyze consequences of these estimates for choice of dictionaries suitable for tasks modeled by the given probabilities. Section 5 is a brief discussion.

2 Approximate measures of network sparsity

We investigate computation of classification tasks represented by binary-valued functions on finite domains $X \subset \mathbb{R}^d$. We denote by

$$\mathscr{B}(X) := \{f \,|\, f : X \to \{-1,1\}\}$$

the *set of all functions on X with values in* $\{-1,1\}$ and by

$$\mathscr{F}(X) := \{f \,|\, f : X \to \mathbb{R}\}$$

the *set of all real-valued functions on X*.

When $\mathrm{card}X = m$ and $X = \{x_1,\ldots,x_m\}$ is a linear ordering of X, then the mapping $\iota : \mathscr{F}(X) \to \mathbb{R}^m$ defined as $\iota(f) := (f(x_1),\ldots,f(x_m))$ is an isomorphism. So, on $\mathscr{F}(X)$ we have the Euclidean inner product defined as

$$\langle f,g\rangle := \sum_{u\in X} f(u)g(u)$$

and the Euclidean norm $\|f\| := \sqrt{\langle f,f\rangle}$. We consider binary-valued functions with the range $\{-1,1\}$ instead of $\{0,1\}$ as all functions in $\mathscr{B}(X)$ have norms equal to $\sqrt{\mathrm{card}X}$.

A *feedforward network with a single linear output* can compute input-output functions from the set

$$\mathrm{span}\, G := \left\{ \sum_{i=1}^{n} w_i g_i \,\middle|\, w_i \in \mathbb{R},\, g_i \in G,\, n \in \mathbb{N} \right\},$$

where G, called a *dictionary*, is a parameterized family of functions. In networks with one hidden layer (called *shallow networks*), G is formed by functions computable by a given type of computational units, whereas in networks with several hidden layers (called *deep networks*), it is formed by combinations and compositions of functions representing units from lower layers (see, e.g., [2, 16]).

Formally, the number of hidden units in a shallow network or in the last hidden layer of a deep one can be described as the *number of nonzero entries* of the vector of output weights of the network. In applied mathematics, the number of nonzero entries of a vector $w \in \mathbb{R}^n$, denoted $\|w\|_0$, is called *"l_0-pseudonorm"* as it satisfies the equation

$$\|w\|_0 = \sum_{i=1}^{n} w_i^0.$$

The quotation marks are used because $\|w\|_0$ is neither a norm nor a pseudonorm. Minimization of "l_0-pseudonorm" is a difficult non convex problem as l_0 lacks the homogeneity property of a norm and its "unit ball" is not convex.

Instead of the nonconvex l_0-functional, its approximation by the *l_1-norm*

$$\|w\|_1 = \sum_{i=1}^{n} |w_i|$$

have been used as a stabilizer in weight-decay regularization methods [7]. Some insight into efficiency of computation of a function f by networks with units from a dictionary G can be obtained from investigation of the minima of l_1-norms of all vectors from the set

$$W_f(G) = \{w = (w_1,\ldots,w_n) \,|\, f = \sum_{i=1}^{k} w_i g_i, g_i \in G, n \in \mathbb{N}\}.$$

Minima of l_1-norms of elements of $W_f(G)$ are bounded from below by a norm of f tailored to a dictionary G called *G-variation*. It is defined for a bounded subset G of a normed linear space $(\mathscr{X}, \|.\|)$ as

$$\|f\|_G := \inf\left\{ c \in \mathbb{R}_+ \;\middle|\; f/c \in \mathrm{cl}_{\mathscr{X}}\, \mathrm{conv}\,(G \cup -G) \right\},$$

where $-G := \{-g \mid g \in G\}$, $\mathrm{cl}_{\mathscr{X}}$ denotes the closure with respect to the topology induced by the norm $\|\cdot\|_{\mathscr{X}}$, and conv denotes the convex hull. Variation with respect to Heaviside perceptrons (called *variation with respect to half-spaces*) was introduced in [1] and extended to general dictionaries in [9].

It is easy to check (see [13]) that for a finite dictionary G and any f, such that the set $W_f(G)$ non empty, G-variation of f is equal to the minimum of l_1-norms of output-weight vectors of shallow networks with units from G, which compute f, i.e.,

$$\|f\|_G = \min\left\{\|w\|_1 \,\middle|\, w \in W_f(G)\right\}.$$

Thus lower bounds on minima of l_1-norms of output-weight vectors of networks computing a function f can be obtained from lower bounds on variational norms. Such bounds can be derived using the following theorem, which is a special case of a more general result [10] proven using Hahn-Banach theorem. By $G^\perp$ is denoted the *orthogonal complement of* G in the Hilbert space $\mathscr{F}(X)$.

Theorem 1. *Let X be a finite subset of $\mathbb{R}^d$ and G be a bounded subset of $\mathscr{F}(X)$. Then for every $f \in \mathscr{F}(X) \setminus G^\perp$,*

$$\|f\|_G \geq \frac{\|f\|^2}{\sup_{g\in G}|\langle g, f\rangle|}.$$

So functions which are nearly orthogonal to all elements of a dictionary G have large G-variations. On the other hand, if a function is correlated with some element of G, then it is close to this element and so can be well approximated by an element of G.

3 Probabilistic bounds

When we do not have any prior knowledge about a type of classification tasks to be computed, we have to assume that a network from the class has to be capable to compute any uniformly randomly chosen function on a given domain. Often in practical applications, most of the binary-valued functions o a given domain are not likely to represent tasks of interest. In such cases some knowledge is available that can be expressed in terms of a discrete probability measure on the set of all functions on X.

For a finite domain $X = \{x_1, \ldots, x_m\} \subset \mathbb{R}^d$, a function f in $\mathscr{B}(X)$ can be represented as a vector $(f(x_1), \ldots, f(x_m)) \in \{-1, 1\}^m \subset \mathbb{R}^m$. We assume that for each $x_i \in X$, there exists a known probability $p_i \in [0,1]$ that $f(x_i) = 1$. For $p = (p_1, \ldots, p_m)$, we denote by

$$\rho_p : \mathscr{B}(X) \to [0,1]$$

the *product probability* defined for every $f \in \mathscr{B}(X)$ as

$$\rho_p(f) := \prod_{i=1}^m \rho_{p,i}(f), \tag{1}$$

where $\rho_{p,i}(f) := p_i$ if $f(x_i) = 1$ and $\rho_{p,i}(f) := 1 - p_i$ if $f(x_i) = -1$. It is easy to verify that ρ_p is a probability measure on $\mathscr{B}(X)$.

When $\mathrm{card} X$ is large, the set $\mathscr{F}(X)$ is isometric to a high-dimensional Euclidean space and $\mathscr{B}(X)$ to a high-dimensional Hamming cube. In high-dimensional spaces and cubes various concentration of measure phenomena occur [14]. We apply the Chernoff-Hoeffding Bound on sums of independent random variables, which do not need to be identically distributed [5, Theorem 1.11] to obtain estimates of distributions of inner products of any fixed function $h \in \mathscr{B}(X)$ with functions randomly chosen from $\mathscr{B}(X)$ with probability ρ_p.

Theorem 2 (Chernoff-Hoeffding Bound)**.** *Let m be a positive integer, $Y_1, \ldots, Y_m$ independent random variables with values in real intervals of lengths $c_1, \ldots, c_m$, respectively, $\varepsilon > 0$, and $Y := \sum_{i=1}^m Y_i$. Then*

$$\Pr\left(|Y - E(Y)| \geq \varepsilon\right) \leq e^{-\frac{2\varepsilon^2}{\sum_{i=1}^m c_i^2}}.$$

For a function $h \in \mathscr{B}(X)$ and $p = (p_1, \ldots, p_m)$, where $p_i \in [0,1]$, we denote by

$$\mu(h,p) := E_p\big(\langle h, f\rangle \,|\, f \in \mathscr{B}(X)\big)$$

the *mean value of inner products* of h with f randomly chosen from $\mathscr{B}(X)$ with probability ρ_p, and by $h^o := \frac{h}{\|h\|}$ its normalization. The next theorem estimates the distribution of these inner products.

Theorem 3. *Let $X = \{x_1, \ldots, x_m\} \subset \mathbb{R}^d$, $p = (p_1, \ldots, p_m)$ be such that $p_i \in [0,1]$, $i = 1, \ldots, m$, and $h \in \mathscr{B}(X)$. Then the inner product of h with f randomly chosen from $\mathscr{B}(X)$ with a probability $\rho_p(f)$ satisfies for every $\lambda > 0$*

i) $$\Pr\Big(|\langle f, h\rangle - \mu(h,p)| > m\lambda\Big) \leq e^{-\frac{m\lambda^2}{2}};$$

ii) $$\Pr\Big(|\langle f^\circ, h^\circ\rangle - \frac{\mu(h,p)}{m}| > \lambda\Big) \leq e^{-\frac{m\lambda^2}{2}}.$$

Proof. Let $F_h : \mathscr{B}(X) \to \mathscr{B}(X)$ be an operator composed of sign-flips mapping h to the constant function equal to 1, i.e., $F_h(h)(x_i) = 1$ for all $i = 1, \ldots, m$ and for all $f \in \mathscr{F}(X)$ and all $i = 1, \ldots, m$, $F_h(f)(x_i) = f(x_i)$ if $h(x_i) = 1$ and $F_h(f)(x_i) = -f(x_i)$ if $h(x_i) = -1$. Let $p(h) = (p(h)_1, \ldots, p(h)_m)$ be defined as $p(h)_i = p_i$ if $h(x_i) = 1$ and $p(h)_i = 1 - p_i$ if $h(x_i) = -1$. The inverse operator F_h^{-1} maps the random variable $F_h(f) \in \mathscr{B}(X)$ such that

$$\Pr\left(F_h(f)(x_i) = 1\right) = p(h)_i$$

to the random variable $f \in \mathscr{B}(X)$ such that

$$\Pr(f(x_i)=1)=p_i.$$

Since the inner product is invariant under sign flipping, for every $f\in\mathscr{B}(X)$ we have $\langle f,h\rangle=\langle F_h(f),(1,\dots,1)\rangle=\sum_{i=1}^m F_h(f)(x_i)$. Thus the mean value of the sum of random variables $\sum_{i=1}^m F_h(f)(x_i)$ is $\mu(h,p)$. Applying to this sum the Chernoff-Hoeffding Bound stated in Theorem 2 with $c_1=\cdots=c_m=2$ and $\varepsilon=m\lambda$, we get

$$\Pr\Big(|\sum_{i=1}^m F_h(f)(x_i)-\mu(h,p)|>m\lambda\Big)\le e^{-\frac{m\lambda^2}{2}}.$$

Hence

$$\Pr\Big(|\langle f,h\rangle-\mu(h,p)|>m\lambda\Big)\le e^{-\frac{m\lambda^2}{2}},$$

which proves i).

ii) follows from i) as all functions in $\mathscr{B}(X)$ have norms equal to $\sqrt{m}$. □

Theorem 3 shows that when the domain X is large, most inner products of any given function with functions randomly chosen from $\mathscr{B}(X)$ with a probability ρ_p are concentrated around their mean values. For example, setting $\lambda=m^{-1/4}$, we get $e^{-\frac{m\lambda^2}{2}}=e^{-\frac{m^{-1/2}}{2}}$ which is decreasing exponentially fast with increasing size m of the domain.

4 Dictionaries for efficient classification

Theorem 3 implies that when a dictionary G contains a function h, for which the mean value $\mu(h,p)$ is large, then most functions randomly chosen with respect to the probability distribution ρ_p are correlated with h. Thus most classification tasks characterized by ρ_p can be well approximated by a network with just one element h. A dictionary G is also suitable for a given task when such function h can be well approximated by a small network with units from G.

It is easy to calculate the mean value $\mu(h,p)$ of inner products of a fixed function h from $\mathscr{B}(X)$ with randomly chosen functions from $\mathscr{B}(X)$ with respect to the probability ρ_p.

Proposition 4. *Let $h\in\mathscr{B}(X)$ and $p=(p_1\dots,p_m)$, where $p_i\in[0,1]$ for each $i=1,\dots,m$. Then for a function f randomly chosen in $\mathscr{B}(X)$ according to ρ_p, the mean value of $\langle f,h\rangle$ satisfies*

$$\mu(h,p)=\sum_{i\in I_h}(2p_i-1)+\sum_{i\in J_h}(1-2p_i),$$

where $I_h=\{i\in\{1,\dots,m\}\,|\,h(x_i)=1\}$ and $J_h=\{i\in\{1,\dots,m\}\,|\,h(x_i)=-1\}$.

By Theorem 1, variation with respect to a dictionary of a function is large when the function is nearly orthogonal to all elements of the dictionary. For $G:=\{g_1,\dots,g_k\}$, we define

$$\mu_G(p):=\max_{g_i,\dots,g_k}|\mu(g_i,p)|.$$

The next theorem estimates probability distributions of variational norms in dependence on the size of a dictionary.

Theorem 5. *Let $X=\{x_1,\dots,x_m\}\subset\mathbb{R}^d$, $G=\{g_1,\dots,g_k\}\subset\mathscr{B}(X)$, and $p=(p_1,\dots,p_m)$ such that $p_i\in[0,1]$, $i=1,\dots,m$. Then for every $f\in\mathscr{B}(X)$ randomly chosen according to ρ_p and every $\lambda>0$*

$$\Pr\left(\|f\|_G\ge\frac{m}{\mu_G(p)+m\lambda}\right)>1-ke^{-\frac{m\lambda^2}{2}}.$$

Proof. By Theorem 3 (i), we get

$$\Pr\Big(|\langle f,h\rangle-\mu(h,p)|>m\lambda\ \ \forall h\in G\Big)\le ke^{-\frac{m\lambda^2}{2}}.$$

Hence,

$$\Pr\Big(|\langle f,h\rangle-\mu(h,p)|\le m\lambda\ \ \forall h\in G\Big)>1-ke^{-\frac{m\lambda^2}{2}}.$$

As $|\langle f,h\rangle-\mu(h,p)|\le m\lambda$ implies $|\langle f,h\rangle|\le\mu(h,p)+m\lambda$, we get

$$\Pr\Big(|\langle f,h\rangle|\le\mu(h,p)+m\lambda\ \ \forall h\in G\Big)>1-ke^{-\frac{m\lambda^2}{2}}.$$

So by Theorem 1

$$\Pr\left(\|f\|_G\ge\frac{m}{\mu(h,p)+m\lambda}\ \ \forall h\in G\right)>1-ke^{-\frac{m\lambda^2}{2}}.$$

Since by the definition, for every $h\in G$ one has $\mu_G(p)\ge\mu(h,p)$, we obtain

$$\frac{m}{\mu_G(p)+m\lambda}\le\frac{m}{\mu(h,p)+m\lambda}$$

and so

$$\Pr\left(\|f\|_G\ge\frac{m}{\mu_G(p)+m\lambda}\right)>1-ke^{-\frac{m\lambda^2}{2}}.$$

□

Theorem 5 shows that when for all computational units h in a dictionary G, the mean values $\mu(h,p)$ are small, then for large m almost all functions randomly chosen according to ρ_p are nearly orthogonal to all elements of the dictionary G. For example, setting $\lambda=m^{-1/4}$, we get a probability greater than $1-ke^{-\frac{m^{1/2}}{2}}$ that a randomly chosen function has G-variation greater or equal to $\frac{m}{\mu_G(p)+m^{3/4}}$.

Thus when for large m, $\frac{\mu_G(p)}{m}$ is small, G-variation of most functions is large unless the size k of a dictionary G outweighs the factor $e^{-\frac{m\lambda^2}{2}}$.

Function with large G-variations cannot be computed by networks that have both the number of hidden units and all absolute vales of output weights small.

Corollary 1. *Let $X = \{x_1, \ldots, x_m\} \subset \mathbb{R}^d$, $G = \{g_1, \ldots, g_k\} \subset \mathscr{B}(X)$, and $p = (p_1, \ldots, p_m)$ such that $p_i \in [0,1]$, $i = 1, \ldots, m$. Then for every $f \in \mathscr{B}(X)$ randomly chosen according to ρ_p, and every $\lambda > 0$,*

$$\Pr\left(\min\left\{\|w\|_1 \,|\, w \in W_f(G)\right\} \geq \frac{m}{\mu_G(p) + m\lambda}\right) > 1 - k\, e^{-\frac{m\lambda^2}{2}}.$$

Corollary 1 implies that computation of most classification tasks randomly chosen from $\mathscr{B}(X)$ with the product probability ρ_p either requires to perform an ill-conditioned task by a moderate network or a well-conditioned task by a large network.

In particular, for the uniform distribution $p_i = 1/2$ for all $i = 1, \ldots, m$, for every $h \in \mathscr{B}(X)$ the mean value $\mu(h, p)$ is zero. Thus for any dictionary $G \subset \mathscr{B}(X)$, almost all functions uniformly randomly chosen from $\mathscr{B}(X)$ are nearly orthogonal to all elements of the dictionary. So we get the following two corollaries.

Corollary 2. *Let $X = \{x_1, \ldots, x_m\} \subset \mathbb{R}^d$ and $f \in \mathscr{B}(X)$ be uniformly randomly chosen. Then for every $h \in \mathscr{B}(X)$ and every $\lambda > 0$*

$$\Pr\left(|\langle f, h\rangle| > m\lambda\right) \leq e^{-\frac{m\lambda^2}{2}}.$$

Corollary 3. *Let $X = \{x_1, \ldots, x_m\} \subset \mathbb{R}^d$ and $G = \{g_1, \ldots, g_k\} \subset \mathscr{B}(X)$. Then for every $f \in \mathscr{B}(X)$ uniformly randomly chosen and every $\lambda > 0$*

$$\Pr\left(\|f\|_G \geq \frac{1}{\lambda}\right) \geq 1 - k\, e^{-\frac{m\lambda^2}{2}}.$$

When we do not have any a priori knowledge about the task, we have to assume that the probability on $\mathscr{B}(X)$ is uniform. Corollary 3 shows that unless a dictionary G is sufficiently large to outweigh the factor $e^{-\frac{m\lambda^2}{2}}$, most functions randomly chosen in $\mathscr{B}(X)$ according to ρ_p have G-variations greater or equal to $1/\lambda$. So for small λ and sufficiently large m, most such functions cannot be computed by linear combinations of small numbers of elements of G with small coefficients. Similar situation occurs when probabilities are nearly uniform.

Many common dictionaries used in neurocomputing are relatively small with respect to the factor $e^{-\frac{m^{1/2}}{2}}$. For example, the size of the *dictionary of signum perceptrons* $P_d(X)$ on a set X of m points in $\mathbb{R}^d$ is well-known since the work of Schläfli [18]. He estimated the number of linearly separated dichotomies of m points in $\mathbb{R}^d$. His upper bound states that for every $X \subset \mathbb{R}^d$ such that $\operatorname{card} X = m$,

$$\operatorname{card} P_d(X) \leq 2\sum_{l=1}^{d}\binom{m-1}{l} \leq 2\frac{m^d}{d!}. \qquad (2)$$

(see, e.g., [4]). The set $P_d(X)$ forms only a small fraction of the set of all functions in the set $\mathscr{B}(X)$, whose cardinality is equal to 2^m. Also other dictionaries of $\{-1,1\}$-valued functions generated by dichotomies of m points in $\mathbb{R}^d$ defined by nonlinear separating surfaces (such as hyperspheres or hypercones) are relatively small (see [4, Table I]).

5 Discussion

As the number of binary-valued functions modeling classification tasks grows exponentially with the size of their domains, we proposed to model relevance of such tasks for a give application area by a probabilistic model. For sets of classification tasks endowed with product probability distributions, we investigated complexity of networks computing these tasks. We explored network complexity in terms of approximate measures of sparsity formalized by l_1 and variational norms. For functions on large domains, we analyzed implications of the concentration of measure phenomena for correlations between network units and randomly chosen functions.

We focused on classification tasks characterized by product probabilities. To derive estimates of complexity of networks computing randomly chosen functions we used the Chernoff-Hoeffding Bound on sums of independent random variables. An extension of our analysis to tasks characterized by more general probability distributions is a subject of our future work. To obtain estimates for more general probability distributions, we plan to apply versions of the Chernoff-Hoeffding Bound stated in [6], which hold in situations when random variables are not independent .

Acknowledgments. V. K. was partially supported by the Czech Grant Foundation grant GA18-23827S and institutional support of the Institute of Computer Science RVO 67985807. M. S. was partially supported by a FFABR grant of the Italian Ministry of Education, University and Research (MIUR). He is Research Associate at INM (Institute for Marine Enginering) of CNR (National Research Council of Italy) under the Project PDGP 2018/20 DIT.AD016.001 "Technologies for Smart Communities" and he is a member of GNAMPA-INdAM (Gruppo Nazionale per l'Analisi Matematica, la Probabilità e le loro Applicazioni - Instituto Nazionale di Alta Matematica).

References

[1] A. R. Barron. Neural net approximation. In K. S. Narendra, editor, *Proc. 7th Yale Workshop on Adaptive and Learning Systems*, pages 69–72. Yale University Press, 1992.

[2] Y. Bengio and A. Courville. Deep learning of representations. In *Handbook of Neural Information Processing*. M. Bianchini and M. Maggini and L. Jain, Berlin, Heideleberg, 2013.

[3] Y. Bengio, O. Delalleau, and N. Le Roux. The curse of highly variable functions for local kernel machines. In *Advances in Neural Information Processing Systems*, volume 18, pages 107–114. MIT Press, 2006.

[4] T. Cover. Geometrical and statistical properties of systems of linear inequalities with applictions in pattern recognition. *IEEE Trans. on Electronic Computers*, 14:326–334, 1965.

[5] B. Doerr. Analyzing randomized search heuristics: Tools from probability theory. In *Theory of Randomized Seach Heuristics - Foundations and Recent Developments*, chapter 1, pages 1–20. World Scientific Publishing, 2011.

[6] D. P. Dubhashi and A. Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, New York, 2009.

[7] T. L. Fine. *Feedforward Neural Network Methodology*. Springer, Berlin Heidelberg, 1999.

[8] Y. Ito. Finite mapping by neural networks and truth functions. *Mathematical Scientist*, 17:69–77, 1992.

[9] V. Kůrková. Dimension-independent rates of approximation by neural networks. In K. Warwick and M. Kárný, editors, *Computer-Intensive Methods in Control and Signal Processing. The Curse of Dimensionality*, pages 261–270. Birkhäuser, Boston, MA, 1997.

[10] V. Kůrková. Complexity estimates based on integral transforms induced by computational units. *Neural Networks*, 33:160–167, 2012.

[11] V. Kůrková and M. Sanguineti. Approximate minimization of the regularized expected error over kernel models. *Mathematics of Operations Research*, 33:747–756, 2008.

[12] V. Kůrková and M. Sanguineti. Model complexities of shallow networks representing highly varying functions. *Neurocomputing*, 171:598–604, 2016.

[13] V. Kůrková, P. Savický, and K. Hlaváčková. Representations and rates of approximation of real-valued Boolean functions by neural networks. *Neural Networks*, 11:651–659, 1998.

[14] M. Ledoux. *The Concentration of Measure Phenomenon*. AMS, Providence, 2001.

[15] H.W. Lin, M. Tegmark, and D. Rolnick. Why does deep and cheap learning work so well? *J. of Statistical Physics*, 168:1223–1247, 2017.

[16] H. N. Mhaskar and T. Poggio. Deep vs. shallow networks: An approximation theory perspective. *Analysis and Applications*, 14:829–848, 2016.

[17] Y. Plan and R. Vershynin. One-bit compressed sensing by linear programming. *Communications in Pure and Applied Mathematics*, 66:1275–1297, 2013.

[18] L. Schläfli. *Theorie der Vielfachen Kontinuität*. Zürcher & Furrer, Zürich, 1901.

[19] T.Hastie, R.Tibshirani, and M. Wainwright. *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman & Hall/CRC, London, 2015.

[20] A.M. Tillmann. On the computational intractability of exact and approximate dictionary learning. *IEEE Signal Processing Letters*, 22:45–49, 2015.

S. Krajči (ed.): ITAT 2018 Proceedings, pp. 92–99
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Jiří Kožusznik, Petr Pulc, and Martin Holeňa

Sentiment Analysis from Utterances

Jiří Kožusznik[1], Petr Pulc[1,2], Martin Holeňa[2]

[1] Faculty of Information Technology, Czech Technical University, Thákurova 7, Prague,Czech Republic
[2] Institute of Computer Science, Czech Academy of Sciences, Pod vodárenskou věží 2, Prague, Czech Republic

Abstract: The recognition of emotional states in speech is starting to play an increasingly important role. However, it is a complicated process, which heavily relies on the extraction and selection of utterance features related to the emotional state of the speaker. In the reported research, MPEG-7 low level audio descriptors[10] serve as features for the recognition of emotional categories. To this end, a methodology combining MPEG-7 with several important kinds of classifiers is elaborated.

1 Introduction

The recognition of emotional states in speech is expected to play an increasingly important role in applications such as media retrieval systems, car management systems, call center applications, personal assistants and the like. In many languages it is common that the meaning of spoken words changes depending on speakers emotions, and consequently the emotional information is important in order to understand the intended meaning. Emotional Speech recognition is a complicated process. Its performance heavily relies on the extraction and selection of features related to the emotional state of the speaker in the audio signal of an utterance. For most of them, the methodology has already been implemented, and they have been experimentally tested and compared Berlin database of emotional speech.

In the reported work in progress, we use MPEG-7 low level audio descriptors[10] as features for the recognition of emotional categories. To this end, we elaborate a methodology combining MPEG-7 with several important kinds of classifiers. For most of them, the methodology has already been implemented and tested with the publicly available Berlin Database of Emotional Speech [1].

In the next section, the task of sentiment analysis from utterances is briefly sketched. Section 3 recalls the necessary background concerning MPEG-7 audio descriptors and the considered classification methods. In Section 4, the principles of the proposed approach are explained. Finally, Section 5 presents results of experimental testing and comparison of the already implemented classifiers on the publicly available Berlin database of emotional speech.

2 Sentiment Analysis from Utterances

Due to the importance of recognizing emotional states in speech, research into sentiment analysis from utterances has been emerging during recent years. We are aware of 3 publications reporting research with the same database of emotional utterances as we used – the Berlin Database of Emotional Speech, used in our research. Let us recall each of them.

The research most similar to ours has been reported in [12], where the authors also used MPEG-7 descriptors for sentiment analysis from utterance. However, they used only scalar MPEG-7 descriptors or scalars derived with time-series descriptors using the software tools Sound Description Toolbox [13] and MPEG-7 Audio Reference Software Toolkit[2], whereas we are implementing also a long-short-term memory network that will use directly the time series. They also used only one classifer in their experiments, a combination of a radial basis function network and a support vector machine.

In [11], emotions are recognized using pitch and prosody features, which are mostly in time domain. Also in that paper, the experiments were performed, and the authors used only one classifer, this time a support vector machine (SVM).

The authors of [16] proposed a set of new 68 features, such as some new based on harmonic frequencies or on the Zipf distribution, for better speech emotion recognition. This set of features is used in a multi-stage classification. When performing the sentiment analysis of the Berlin Database, the utterance classification to the considered emotional categories was preceded with a gender classification of the speakers, and the gender of the speaker was subsequently used as an additional feature for the classification of the utterances.

3 MPEG-7 Audio Descriptors

MPEG-7 is a standard for low-level description of audio signals, describing a signal by means of the following groups of descriptors[10]:

1. Basic: Audio Power (AP), Audio Waveform(AWF). Temporally sampled scalar values for general use, applicable to all kinds of signals. The AP describes the temporally-smoothed instantaneous power of samples in the frame,in other words it is a temporally measure of signal content as a function of time and offers a quick summary of a signal in conjunction with other basic spectral descriptors. The AWF describes audio waveform envelope (minimum and maximum), typically for display purposes.

2. Basic Spectral: Audio Spectrum Envelop (ASE), Audio Spectrum Centroid (ASC), Audio Spectrum Spread (ASS), Audio Spectrum Flatness (ASF).
   All share a common basis, all deriving from the short term audio signal spectrum (analysis of frequency over time). They are all based on the ASE Descriptor, which is a logarithmic-frequency spectrum. This descriptor provides a compact description of the signal spectral content and represents the similar approximation of logarithmic response of the human ear. The ASE descriptor is an indicator as to whether the spectral content of a signal is dominated by high or low frequencies. The ASC Descriptor could be considered as an approximation of perceptual sharpness of the signal. The ASS descriptor indicates whether the signal content, as it is represented by the power spectrum, is concentrated around its centroid or spread out over a wider range of the spectrum. This gives a measure which allows the distinction of noise-like sounds from tonal sounds. The ASF describes the flatness properties of the spectrum of an audio signal for each of a number of frequency bands.

3. Basic Signal Parameters: Audio Fundamental Frequency (AFF) and Audio Harmonicity (AH).
   The signal parameters constitute a simple parametric description of the audio signal. This group includes the computation of an estimate for the fundamental frequency (F0) of the audio signal. The AFF descriptor provides estimates of the fundamental frequency in segments in which the audio signal is assumed to be periodic. The AH represents the harmonicity of a signal, allowing distinction between sounds with a harmonic spectrum (e.g., musical tones or voiced speech e.g., vowels), sounds with an inharmonic spectrum (e.g., bell-like sounds) and sounds with a non-harmonic spectrum (e.g., noise, unvoiced speech).

4. Temporal Timbral: Log Attack Time (LAT), Temporal Centroid (TC).
   Timbre refers to features that allow one to distinguish two sounds that are equal in pitch, loudness and subjective duration. These descriptors are taking into account several perceptual dimensions at the same time in a complex way. Temporal Timbral descriptors describe the signal power function over time. The power function is estimated as a local mean square value of the signal amplitude value within a running window. The LAT descriptor characterizes the "attack" of a sound, the time it takes for the signal to rise from silence to its maximum amplitude. This feature signifies the difference between a sudden and a smooth sound. The TC descriptor computes a time-based centroid as the time average over the energy envelope of the signal.

5. Timbral Spectral descriptors: Harmonic Spectral Centroid (HSC), Harmonic Spectral Deviation (HSD), Harmonic Spectral Spread (HSS), Harmonic Spectral Variation (HSV) and Spectral Centroid.
   These are spectral features extracted in a linear-frequency space. The HSC descriptor is defined as the average, over the signal duration, of the amplitude-weighted mean of the frequency of the bins (the harmonic peaks of the spectrum) in the linear power spectrum. It is has a high correlation with the perceptual feature of "sharpness" of a sound. The HSD descriptor measures the spectral deviation of the harmonic peaks from the global envelope. The HSS descriptor measures the amplitude-weighted standard deviation (Root Mean Square) of the harmonic peaks of the spectrum, normalized by the HSC. The HSV descriptor is the normalized correlation between the amplitude of the harmonic peaks between two subsequent time-slices of the signal.

6. Spectral Basis, which consists of Audio Spectrum Basis (ASB) and Audio Spectrum Projection (ASP).

3.1 Tools for Working with MPEG-7 Descriptors

We utilized the Sound Description Toolbox [13] and MPEG-7 Audio Analyzer - Low Level Descriptors Extractor [15] for our experiments. Both of them extract a number of MPEG-7 standard descriptors, both scalar ones and time series. In addition, the SDT also calculates perceptual features such as Mel Frequency Cepstral Coefficients, Specific Loudness and Sensation Coefficients. From this descriptors calculate means, covariances, means of first-order differences and covariances of first order differences. The Total number of features provided by this toolbox is 187.

4 Employed Classification Methods

We have elaborated our approach to sentiment analysis from utterances for six classification methods: k nearest neighbors, support vector machines, multilayer perceptrons, classification trees, random forests [7] and long short-term memory (LSTM) network [5, 6, 8]. The first five of them have already been implemented and tested (cf. Section 5), the last and most advanced one is still being implemented.

4.1 k Nearest Neighbours (kNN)

A very traditional way of classifying a new feature vector $x \in \mathscr{X}$ if a sequence of training data $(x_1,c_1),\ldots,(x_p,c_p)$ is available is the nearest neighbour method: take the x_j that is the closest to x among $x_1,\ldots,x_p$, and assign to x the class assigned to x_j, i.e., c_j.

A straightforward generalization of the nearest neighbour method is to take among $x_1,\ldots,x_p$ not one, but k feature vectors $x_{j_j},\ldots,x_{j_k}$ closest to x. Then x is assigned the

class $c \in C$ fulfilling

$$|\{i, 1 \leq i \leq k | c_{j_i} = c\}| = max_{c' \in C} |\{i, 1 \leq i \leq k | c_{j_i} = c'\}|. \quad (1)$$

This method is called, expectedly, k nearest neighbours, or k-NN for short.

4.2 Support Vector Machines (SVM)

Support vector machines are classifiers into two classes. This method attempts to derive from the training data $(x_1, c_1), \ldots, (x_p, c_p)$ the best possible generalization to unseen feature vectors.

If both classes, more precisely their intersections with the set $\{x_1, \ldots, x_p\}$ of training inputs, are in the space of feature vectors linearly separable, the method constructs two parallel hyperplanes $H_+ = \{x \in \mathbb{R}^n | x^\top w + b_+ = 0\}, H_- = \{x \in \mathbb{R}^n | x^\top w + b_- = 0\}$ such that the training data fulfil

$$c_k = \begin{cases} 1 & \text{if } x^\top w + b_+ \geq 0, \\ \text{-1} & \text{if } x^\top w + b_- \leq 0, \end{cases} k = 1, \ldots, p, \quad (2)$$

$$H_+ \cap \{x_1, \ldots, x_p\} \neq \emptyset, H_- \cap \{x_1, \ldots, x_p\} \neq \emptyset. \quad (3)$$

The hyperplanes H_+ and H_- alle called support hyperplanes. Their common normal vector w and intercepts b_+, b_- are obtained through solving the following constrained optimization task:

Maximize with respect to w, b_+, b_- the distance

$$d(H_+, H_-) = \frac{b_+ - b_-}{\|w\|} \quad (4)$$

on condition that the p inequalities (2) hold.

The distance (4) is commonly called margin. The solution to this optimization task coincides with the $(w^*, b_+^*, b_-^*, \alpha_1^*, \ldots, \alpha_p^*)$ of the Lagrange function

$$L(w, b_+, b_-, \alpha_1, \ldots, \alpha_p) = \|w\|^2 + \sum_{k=1}^{p} \alpha_k (\frac{, b_+ - b_-}{2} - c_k x_k^\top w) \quad (5)$$

where $\alpha_1, \ldots, \alpha_p \geq 0$ are Lagrange coefficients of the optimization task. Once the saddle point $(w^*, b_+^*, b_-^*, \alpha_1^*, \ldots, \alpha_p^*)$ is found, the classifier is defined by

$$\phi(x) = \begin{cases} 1 & \text{if } \sum_{x_k \in \mathscr{S}} \alpha_k^* c_k x^\top x_k + b^* \geq 0, \\ -1 & \text{if } \sum_{x_k \in \mathscr{S}} \alpha_k^* c_k x^\top x_k + b^* < 0, \end{cases} \quad (6)$$

where $b^* = \frac{1}{2}(b_+^* + b_-^*)$ and

$$\mathscr{S} = \{x_k | \alpha_k^* > 0\}. \quad (7)$$

Due to the Karush-Kuhn-Tucker (KKT) conditions,

$$\alpha_k^* (\frac{b_+^* - b_-^*}{2} - c_k x_k^\top w^*) = 0, k = 1, \ldots, p, \quad (8)$$

all feature vectors from the set $\mathscr{S}$ lie on some of the suport hyperplanes (3). Therefore, they are called support vectors. This name together with the observation that they completely determine the classifier defined in (6) explains why such a classifier is called support vector machine.

If the intersections of both classes with the training inputs are not linearly separable, a SVM is constructed similarly, but instead of the set of possible fature vectors, now the set of functions

$$\kappa(\cdot, x) \text{ for all possible feature vectors } x \quad (9)$$

is considered, where κ is a kernel, i.e., a mapping on pairs of feature vectors that is symmetric and such that for any $k \in \mathbb{N}$ and any sequence of different feature vectors $x_1, \ldots, x_k$, the matrix

$$G_\kappa(x_1, \ldots, x_k) = \begin{pmatrix} \kappa(x_1, x_1) & \ldots & \kappa(x_1, x_k) \\ \ldots\ldots\ldots\ldots\ldots\ldots & & \\ \kappa(x_k, x_1) & \ldots & \kappa(x_k, x_k) \end{pmatrix}, \quad (10)$$

which is called the Gramm matrix of $x_1, \ldots, x_k$, is positive semidefinite, i.e.,

$$(\forall y \in \mathbb{R}^k)\ y^\top G_\kappa(x_1, \ldots, x_k) y \geq 0. \quad (11)$$

The most commonly used kinds of kernels are the Gaussian kernel with a parameter $\varsigma > 0$,

$$(\forall x, x' \in \mathbb{R}^{n'})\ \kappa(x, x') = \exp\left(-\frac{1}{\varsigma}\|x - x'\|^2\right), \quad (12)$$

and polynomial kernel with parameters $d \in \mathbb{N}$ and $c \geq 0$,

$$(\forall x, x' \in \mathbb{R}^{n'})\ \kappa(x, x') = (x^\top x' + c)^d. \quad (13)$$

It is known [14] that, due to the properties of kernels, if the joint distribution of a sequence of different feature vectors $x_1, \ldots, x_k$ is continuous, then almost surely any proper subset of the set of functions $\{\kappa(\cdot, x_1), \ldots, \kappa(\cdot, x_k)\}$ is in the space of all functions (9) linearly separable from its complement.

However, the featre vectors x and x_k can't be simply replaced by the corresponding functions $\kappa(\cdot, x)$ and $\kappa(\cdot, x_k)$ in the definition (6) of a SVM classifier because a transpose $x^\top$ exists for a finite-dimensional vector, but not a for an infinite-dimensional function. Fortunately, the transpose occurs in (6) only as a part of the scalar product $x^\top x_k$. And a scalar product can be defined also on the space of all functions (9). Namely, the properties of a scalar product has the function that to the pair of functions $(\kappa(\cdot, x), \kappa(\cdot, x')$ assigns the value $\kappa(x, x')$. Using this scalar product in (6), we obtain the following definition of a SVM classifier for linearly non-separable classes:

$$\phi(x) = \begin{cases} 1 & \text{if } \sum_{x_k \in \mathscr{S}} \alpha_k^* c_k \kappa(x, x_k) + b \geq 0, \\ -1 & \text{if } \sum_{x_k \in \mathscr{S}} \alpha_k^* c_k \kappa(x, x_k) + b \geq 0. \end{cases} \quad (14)$$

4.3 Multilayer Perceptrons (MLP)

A multilayer percptron is a mapping ϕ of feature vectors to classes with which a directed graph $G_\phi = (\mathscr{V}, \mathscr{E})$ is associated. Due to the inspiration from biological neural networks, the vertices of G_ϕ are called *neurons* and its edges are called *connections*. In addition, G_ϕ is required to have a layered structure, which means that the set $\mathscr{V}$ of neurons can be decomposed into $L+1$ mutually disjoint layers, $\mathscr{V} = \mathscr{V}_0 \cup \mathscr{V}_1 \cup \cdots \cup \mathscr{V}_L, L \geq 2$, such that

$$(\forall (u,v) \in \mathscr{E})\ u \in \mathscr{V}_i, i = 0, \ldots, L-1\ \&\ v \notin \mathscr{V}_i \Rightarrow v \in \mathscr{V}_{i+1}. \tag{15}$$

The layer $\mathscr{I} = \mathscr{V}_0$ is called input layer of the MLP, the layer $\mathscr{O} = \mathscr{V}_L$ its output layer and the layers $\mathscr{H}_1 = \mathscr{V}_1, \ldots, \mathscr{H}_{L-1} = \mathscr{V}_{L-1}$ its hidden layers.

The purpose of the graph G_ϕ associated with the mapping ϕ is to define a decomposition of ϕ into simple mappings assigned to hidden and output neurons and to connections between neurons (input neurons normally only accept the components of the input, and no mappings are assigned to them). Inspired by biological terminology, mappings assigned to neurons are called *somatic*, those assigned to connections are called *synaptic*.

To each connection $(u,v) \in \mathscr{E}$, the multiplication by a weight $w_{(u,v)}$ is assigne as a synaptic mapping:

$$(\forall \xi \in \mathbb{R})\ f_{(u,v)}(\xi) = w_{(u,v)}\xi. \tag{16}$$

To each hidden neuron $v \in \mathscr{H}_i$, the following somatic mapping is assigned:

$$(\forall \xi \in \mathbb{R}^{|\mathrm{in}(v)|})\ f_v(\xi) = \varphi\Big(\sum_{u \in \mathrm{in}(v)} [\xi]_u + b_v\Big), \tag{17}$$

where $[\xi]_u$ for $u \in \mathrm{in}(v)$ denotes the component of ξ that is the output of the synaptic mapping $f_{u,v}$ assigned to the connection (u,v), $\mathrm{in}(v) = \{u \in \mathscr{V} | (u,v) \in \mathscr{E}\}$ is the input set of v, and $\varphi : \mathbb{R} \to \mathbb{R}$ is called activation function. Though the activation functions, in applications typically sigmoidal functions are used to this end, i.e., functions that are non-decreasing, piecewise continuous, and such that

$$-\infty < \lim_{t \to -\infty} \varphi(t) < \lim_{t \to \infty} \varphi(t) < \infty. \tag{18}$$

The activation functions most frequently encountered in MLPs are:

- the logistic function,

$$(\forall t \in \mathbb{R})\ \varphi(t) = \frac{1}{1+\mathrm{e}^{-t}}; \tag{19}$$

- the hyperbolic tangent,

$$\varphi(t) = \tanh t = \frac{\mathrm{e}^t - \mathrm{e}^{-t}}{\mathrm{e}^t + \mathrm{e}^{-t}}. \tag{20}$$

To an output neuron $v \in \mathscr{O}$, also a somatic mapping of the kind (17) with the activation functions (19) or (20) can be assigned. If it is the case, then the class c predicted for a feature vector x is obtained as $c = \arg\max_i(\phi(x))_i$, where $(\phi(x))_i$ denotes the i-the component of $\phi(x)$. Alternatively the activation function assigned to an output neuron can be the step function, aka Heaviside function

$$\varphi(t) = \begin{cases} 0 & \text{if } t < 0, \\ 1 & \text{if } t \geq 0. \end{cases} \tag{21}$$

In that case, the value $(\phi(x))_c$ already directly indicates whether x belongs to the class c.

4.4 Classification Trees (CT)

A classifier $\phi : \mathscr{X} \to C = \{c_1, \ldots, c_m\}$ is called binary classification tree, if there is a binary tree $T_\phi = (V_\phi, E_\phi)$ with vertices V_ϕ and edges E_ϕ such that:

(i) $V_\phi = \{v_1, \ldots, v_L, \ldots, v_{2L-1}\}$, where $L \geq 2$, v_0 is the root of T_ϕ, $v_1, \ldots, v_{L-1}$ are its forks and $v_L, \ldots, v_{2L-1}$ are its leaves.

(ii) If the children of a fork $v \in \{v_1, \ldots, v_{L-1}\}$ are $v^L \in V_\phi$ (left child) and $v^R \in V_\phi$ (right child) and if $v = v_i, v^L = v_j, v^R = v_k$, then $i < j < k$.

(iii) To each fork $v \in \{v_1, \ldots, v_{L-1}\}$, a predicate φ_v of some formal logic is assigned, evaluated on features of the input vectors $x \in \mathscr{X}$.

(iv) To each leaf $v \in \{v_L, \ldots, v_{2L-1}\}$, a class $c_v \in C$ is assigned.

(v) For each input $x \in \mathscr{X}$, the predicate φ_{v_1} assigned to the root is evaluated.

(vi) If for a fork $v \in \{v_1, \ldots, v_{L-1}\}$, the predicate φ_v evaluates true, then $\phi(x) = c_{v^L}$ in case v^L is already a leaf, and the predicate φ_{v^L} is evaluated in case v^L is still a fork.

(vii) If for a fork $v \in \{v_1, \ldots, v_{L-1}\}$, the predicate φ_v evaluates false, then $\phi(x) = c_{v^R}$ in case v^R is already a leaf, and the predicate φ_{v^R} is evaluated in case v^R is still a fork.

4.5 Random Forests (RF)

Random Forests are ensembles of classifiers in which the individual members are classification trees. They are constructed by bagging, i.e., bootstrap aggregation of individual trees, which consists in training each member of the ensemble with another set of training data, sampled randomly with replacement from the original training pairs $(x_1, c_1), \ldots, (x_p, c_p)$. Typical sizes of random forests encountered in applications are dozens to thousands trees. Subsequently, when new subjects are input to the forest, each tree classifies them separately, according to the leaves at which they end, and the final classification by the forest is obtained by means of an aggregation function. The usual aggregation function of random forests is majority voting, or some of its fuzzy generalizations.

According to which kind of randomness is involved in the costruction of the ensemble, two broad groups of random forests can be differentiated:

1. *Random forests grown in the full input space.* Each tree is trained using all considered input features. Consequently, any feature has to be taken into account when looking for the split condition assigned to an inner node of the tree. However, features actually occurring in the split conditions can be different from tree to tree, as a consequence of the fact that each tree is trained with another set of training data. For the same reason, even if a particular feature occurs in split conditions of two different trees, those conditions can be assigned to nodes at different levels of the tree.

   A great advantage of this kind of random forests is that each tree is trained using all the information available in its set of training data. Its main disadvantage is high computational complexity. In addition, if several or even only one variable are very noisy, that noise gets nonetheless incorporated into all trees in the forest. Because of those disadvantages, random forests are grown in the complete input space primarily if its dimension is not high and no input feature is substantially noisier than the remaining ones.

2. *Random forests grown in subspaces of the input space.* Each tree is trained using only a randomly chosen fraction of features, typically a small one. This means that a tree t is actually trained with projections of the training data into a low-dimensional space spanned by some randomly selected dimensions $i_{t,1} \leq \cdots \leq i_{t,d_t} \in \{1,\ldots,d\}$, where d is the dimension of the input space, and d_t is typically much smaller than d. Using only a subset of features not only makes forest training much faster, but also allows to eliminate noise originating from only several features. The price paid for both these advantages is that training makes use of only a part of the information available in the training data.

4.6 Long Short-Term Memory (LSTM)

An LSTM network is used for classification of sequences of feature vectors, or equivalently, multidimensional time series with discrete time. Alternatively, it can be also employed to obtain sequences of such classifications, i.e., in situations when the neural network input is a sequence of feature vectors and its output is a a sequence of classes. Differently to most of other commonly encountered kinds of artificial neural networks, an LSTM layer connects not simple neurons, but units with their own inner structure. Several variants of an LSTM have been proposed (e.g., [5, 6]), all of them include at least the following four kinds of units described below. Each of them has certain properties of usual ANN neurons, in particular, the values assigned to them depend, apart from a bias, on values assigned to the unit input at the same time step and on values assigned to the unit output at the previous time step. Hence, an LSTM network layers is a recurrent network.

(i) *Memory cells* can store values, aka cell states, for an arbitray time. They have no activation function, thus their output is actually a biased linear combination of unit inputs and of the values from the previous time step coming through recurrent connections.

(ii) *Input gate* controls the extent to which values from the previous unit or from the preceding layer influence the value stored in the memory cell. It has a sigmoidal activation function, which is applied to a biased linear combination of unit inputs and of values from the previous time step, though the bias and synaptic weights of the input and recurrent connections are specific and in general different from the bias and synaptic weights of the memory cell.

(iii) *Forget gate* controls the extent to which the memory cell state is supressed. It again has a sigmoidal activation function, which is applied to a specific biased linear combination of unit inputs and of values from the previous time step.

(iv) *Output gate* controls the extent to which the memory cell state influences the unit output. Also this gate has a sigmoidal activation function, which is applied to a specific biased linear combination of unit inputs and of values from the previous time step, and subsequently composed either directly with the cell state or with its sigmoidal transformation, using another sigmoid than is used by the gates.

5 Experimental Testing

5.1 Berlin Database of Emotional Speech

For the evaluation of already implemented classifiers, we used the publicly available dataset "EmoDB", aka Berlin database of emotional speech. It consists of 535 emotional utterances in 7 emotional categories namely anger, boredom, disgust, fear, happiness, sadness and neutral. These utterances are sentences read by 10 professional actors, 5 males and 5 females [1], which were recorded in an anechoic chamber under supervision by linguists and psychologists) . The actors were advised to read these predefined sentences in the targeted emotional categories, but the sentences do not contain any emotional bias. A human perception test was conducted with 20 persons, different from the speakers, in order to evaluate the quality of the recorded data with respect to recognisability and naturalness of presented emotion. This evaluation yielded a mean accuracy 86% over all emotional categories.

5.2 Experimental Settings

As input features, the outputs from the Sound Description Toolbox were used. Consequently, the input dimension

was 187. The already implemented classifiers were compared by means of a 10-fold cross-validation, using the following settings for each of them:

- For the k nearest neighbors classification, the value $k = 9$ was chosen by a grid method from $\langle 1, 80 \rangle$. This classifer was applied to data normalized to zero mean and unit variance.
- Support vector machines are constructed for each of the 7 considered emotions, to classify between that emotion and all the remaining ones. They employ auto-scaled Gaussian kernels and do not use slack variables.
- The MLP has 1 hidden layer with 70 neurons. Hence, taking into account the input dimension and the number of classes, the overall architecture of the MLP is 187-70-7.
- Classification trees are restricted to have at most 23 leaves. This upper limit was chosen by a grid method from $\langle 1, 50 \rangle$, taking into account the way how classification trees are grown in their Matlab implementation.
- Random forests consist of 50 classification trees, each of them taking over the above restriction. The number of trees was selected by a grid method from 10, 20,. . . ,100.

5.3 Comparison of Already Implemented Classifiers

First, we compared the already implemented classifiers on the whole Berlin database of emotional speech, with respect to accuracy and area under the ROC curve (area under curve, AUC). Since a ROC curve makes sense only for a binary classifier, we computed areas under 7 separate curves corresponding to classifiers classifying always 1 emotion against the rest. The results are presented in Table 1 and in Figure 1. They clearly show SVM as the most promising classifier. It has the highest accuracy, and also the AUC for binary classifiers corresponding to 5 of the 7 classifiers

Then we compared the classifiers separately on the utterances of each of the 10 speakers who created the database. The results are summarized in Table 2 for accuracy and Table 3 for AUC averaged over all 7 emotions. They indicate a great difference between most of the compared classifiers. This is confirmed by the Friedman test of the hypotheses that all classifiers have equal accuracy and equal average AUC. The Friedman test rejected both hypotheses with a high significance: With the Holm correction for simultaneously tested hypotheses [9], the achieved significance level (aka p-value) was $4 \cdot 10^{-6}$. For both hypotheses, posthoc tests according to [3, 4] were performed, testing equal accuracy and equal average AUC between individual pairs of classifiers. For

Table 1: Accuracy and area under curve (AUC) of the implemented classifiers on the whole Berlin database of emotional speech. AUC is measured for binary classification of each of the considered 7 emotions against the rest

Classifier	Accuracy	AUC emotion against the rest		
		Anger	Boredom	Disgust
kNN	0.73	0.956	0.933	0.901
SVM	**0.93**	**0.979**	**0.973**	**0.966**
MLP	0.78	0.977	0.969	0.964
DT	0.59	0.871	0.836	0.772
RF	0.71	0.962	0.949	0.920

Classifier	AUC emotion against the rest			
	Fear	Happiness	Neutral	Sadness
kNN	0.902	0.856	0.962	0.995
SVM	**0.983**	0.904	0.974	**0.997**
MLP	0.969	**0.933**	**0.983**	0.996
DT	0.782	0.683	0.855	0.865
RF	0.921	0.882	0.972	0.992

Table 2: Comparison between pairs of implemented classifiers with respect to accuracy, based on 10 independent parts of the Berlin database of emotional speech corresponding to 10 different speakers. The result in a cell of the table indicates on how many parts the accuracy of the row classifier was higher : on how many parts the accuracy of the column classifier was higher. A result in bold indicates that after the Friedman test rejected the hypothesis of equal accuracy of all classifiers, the post-hoc test according to [3, 4] rejects the hypothesis of equal accuracy of the particular row and column classifiers. All simultaneously tested hypotheses were corrected in accordance with Holm [9]

classifier	kNN	SVM	MLP	DT	RF
kNN		**0:10**	3.5:6.5	9:1	5:5
SVM	**10:0**		10:0	**10:0**	**10:0**
MLP	6.5:3.5	0:10		**10:0**	7:3
DT	1:9	**0:10**	**0:10**		0:10
RF	5:5	**0:10**	3:7	10:0	

the family-wise significance level 5%, they reveal the following Holm-corrected significant differences between individual pairs of classifiers: both for accuracy and averaged AUC: (SVM,DT), (MLP,DT), and in addition between (kNN,SVM), (SVM,RF) for accuracy.

6 Conclusion

The presented work in progress investigated the possibilities to analyse emotions in utterances based on MPEG7 features. So far, we implemented only five classification methods not using time series features, but only 187 scalar features, namely the k nearest neighbours classifier, support vector machines, mutilayer perceptrons, decision trees and random forests. The obtained results in-

Table 3: Comparison between pairs of implemented classifiers with respect to the AUC averaged over all 7 emotions, based on 10 independent parts of the Berlin database of emotional speech corresponding to 10 different speakers. The result in a cell of the table indicates on how many parts the AUC of the row classifier was higher : on how many parts the AUC of the column classifier was higher. A result in bold indicates that after the Friedman test rejected the hypothesis of equal AUC of all classifiers, the post-hoc test according to [3, 4] rejects the hypothesis of equal AUC of the particular row and column classifiers. All simultaneously tested hypotheses were corrected in accordance with Holm [9]

classifier	kNN	SVM	MLP	DT	RF
kNN		2:8	0:10	10:0	4:6
SVM	8:2		5:5	**10:0**	9:1
MLP	10:0	5:5		**10:0**	9:1
DT	0:10	**0:10**	**0:10**		0:10
RF	6:4	1:9	1:9	10:0	

dicate that especially support vector machines and multilayer perceptrons are quite successfull for this task. Statistical testing confirms significant differences between these two kinds of classifiers on the one hand, and decision trees an random forests on the other hand.

The next step in this ongoing research is to implement the long short-term memory neural network, recalled in Subsection 4.6, because they can work not only with scalar features but also with features represented with time series.

Acknowledgement

The research reported in this paper has been supported by the Czech Science Foundation (GAČR) grant 18-18080S.

References

[1] F. Burkhardt, A. Paeschke, M. Rolfes, W. Sendlmeier, and B. Weiss. A database of german emotional speech. In *Interspeech*, pages 1517–1520, 2005.

[2] M. Casey, A. De Cheveigne, P. Gardner, M. Jackson, and G. Peeters. MPEG-7 multimedia software resources. http://mpeg7.doc.gold.ac.uk/, 2001.

[3] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[4] S. Garcia and F. Herrera. An extension on "Statistical Comparisons of Classifiers over Multiple Data Sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694, 2008.

[5] F.A. Gers, J. Schmidhuber, and J. Cummis. Learning to forget: Continual prediction with LSTM. In *9th International Conference on Artificial Neural Networks: ICANN '99*, pages 850–855, 1999.

[6] A. Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*. PhD thesis, TU München, 2008.

[7] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning, 2nd Edition*. Springer, 2008.

[8] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[9] S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6:65–70, 1979.

[10] H.G. Kim, N. Moreau, and T. Sikora. *MPEG-7 Audio and Beyond: Audio Content Indexing and Retrieval*. John Wiley and Sons, New York, 2005.

[11] S. Lalitha, A. Madhavan, B. Bhusan, and S. Saketh. Speech emotion recognition. In *International Conference on Advances in Electronics*, pages 92–95, 2014.

[12] A.S. Lampropoulos and G.A. Tsihrintzis. Evaluation of MPEG-7 descriptors for speech emotional recognition. In *Eighth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pages 98–101, 2012.

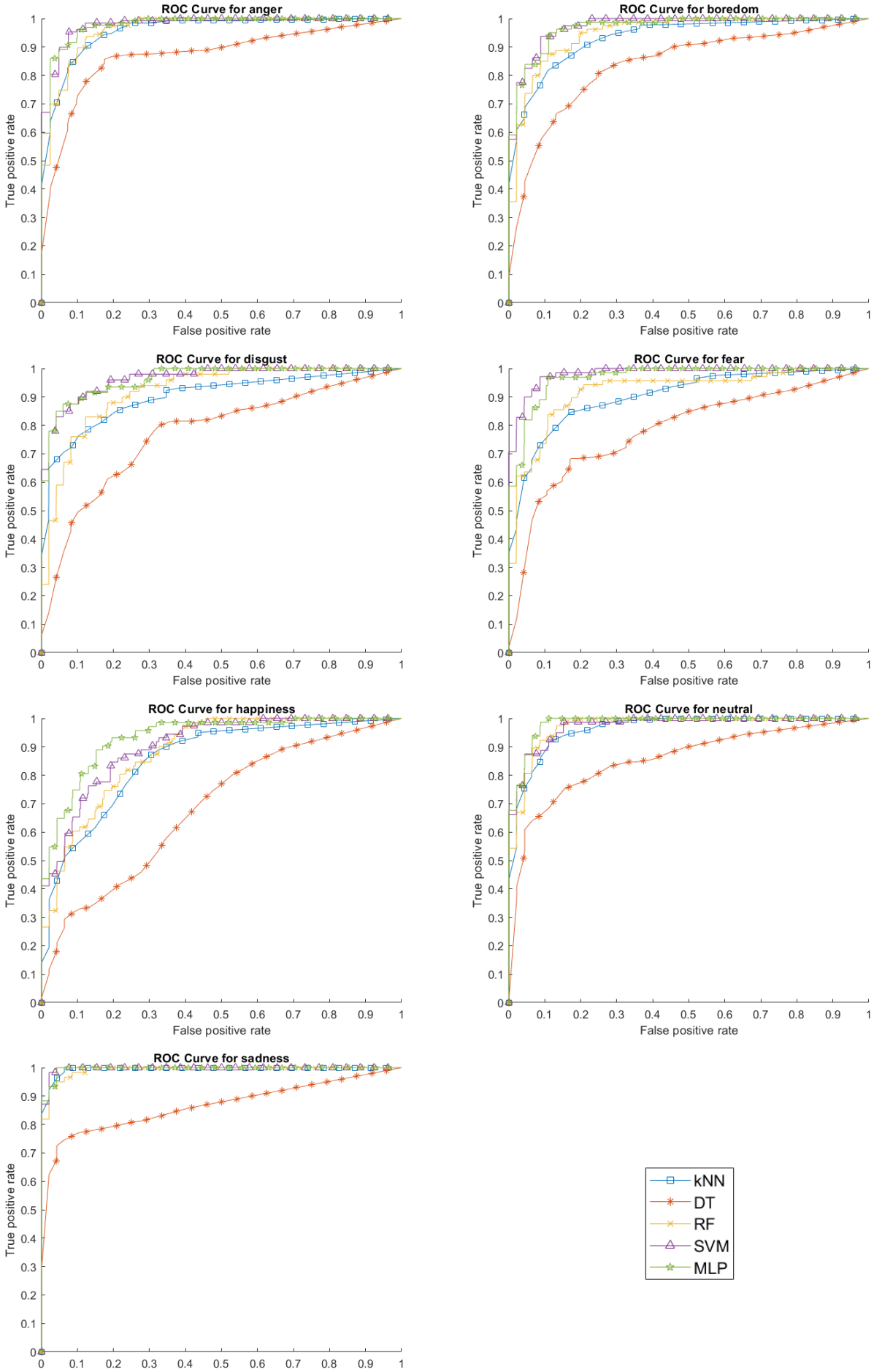
[13] A. Rauber, T. Lidy, J. Frank, E. Benetos, V. Zenz, G. Bertini, T. Virtanen, A.T. Cemgil, S. Godsill, D. Clark, P. Peeling, E. Peisyer, Y. Laprie, A. Sloin, A. Alfandary, and D. Burshtein. MUSCLE network of excellence: Multimedia understanding through semantics, computation and learning. Technical report, TU Vienna, Information and Software Engineering Group, 2004.

[14] B. Schölkopf and A.J. Smola. *Learning with Kernels*. MIT Press, Cambridge, 2002.

[15] T. Sikora, H.G. Kim, N. Moreau, and S. Amjad. MPEG-7-based audio annotation for the archival of digital video. http://mpeg7lld.nue.tu-berlin.de/, 2003.

[16] Z. Xiao, E. Dellandrea, W. Dou, and L. Chen. Multi-stage classification of emotional speech motivated by a dimensional emotion model. *Multimedia Tools and Applicaions*, 46:119–145, 2010.

Figure 1: ROC curve for all emotions on the whole Berlin database



S. Krajči (ed.): ITAT 2018 Proceedings, pp. 100–107
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Oliver Keruľ-Kmec, Petr Pulc, and Martin Holeňa

Semisupervised Segmentation of UHD Video

Oliver Keruľ-Kmec[1], Petr Pulc[1,2], Martin Holeňa[2]

[1] Faculty of Information Technology, Czech Technical University, Thákurova 7, Prague, Czech Republic
[2] Institute of Computer Science, Czech Academy of Sciences, Pod vodárenskou věží 2, Prague, Czech Republic

Abstract: One of the key preprocessing tasks in information retrieveal from video is the segmentation of the scene, primarily its segmentation into foreground objects and the background. This is actually a classification task, but with the specific property that it is very time consuming and costly to obtain human-labelled training data for classifier training. That suggests to use semisupervised classifiers to this end. The presented work in progress reports the investigation of semisupervised classification methods based on cluster regularization and on fuzzy c-means in connection with the foreground / background segmentation task. To classify as many video frames as possible using only a single human-based frame, the semisupervised classification is combined with a frequently used keypoint detector based on a combination of a corner detection method with a visual descriptor method. The paper experimentally compares both methods, and for the first of them, also classifiers with different delays between the human-labelled video frame and classifier training.

1 Introduction

For the indexing of multimedial content, it is beneficial to have annotations of actors, objects or any other information that can occur in a video. A vital preprocessing task to prepare such annotations is the segmentation of the scene into foreground objects and the background.

Traditional methods, such as Gaussian mixture modeling, work on the pixel level and are time consuming on higher resolution video [1]. Another simple method models the background through image averaging, however it requires a static camera [6]. Our approach, on the other hand, is based on the level of detected interest points, and uses semi-supervised classification to assign those points as belonging either to the foreground objects or to the background.

In the next section, we introduce the key points detector we employed for the detection of points of interest. Section 3 recalls two methods of semi-supervised classification we used in our approach. The approach itself is outlined in Section 4. Finally, Section 5 presents the results of its experimental validation performed so far.

2 Scene Segmentation in the Context of Video Preprocessing

In each frame of the video, a keypoint detector is used to detect points of interest and compute their descriptors. In our research, a combination of a corner detection method FAST (Features from Accelerated Segment Test) with a visual descriptor method BRIEF (Binary Robust Independent Elementary Features) is used to this end, known as ORB (oriented FAST and rotated BRIEF) [7]. Points of interest detected in a frame are always attempted to match those detected in the next frame. Such matching points are searched in a two-step fashion:

(i) Only the points of interest in the spacial neighbourhood of the expected position are considered. That position is based on last known interest point position and its past motion (if available).

(ii) Among the points of interest resulting from (i), as well as among all detected in the current frame for which no information about their past motion is available, points in the previous frame are searched based on the Hamming distance between the descriptors of both points.

Whereas the dependence of matching success on the difference between positions of the points and on the movement of the first point has a straightforward geometric meaning, its dependence on the Hamming distance between their descriptors has a probabilistic character. In [7], this dependence was investigated and was found that if the Hamming distance between 256-bit binary descriptors of the points is greater than 64, then the probability of successful match is less than 5%.

If two points of interests in subsequent frames are considered matching, the point in the later frame is added to the history vector of the point in the previous frame. In this way, we get the motion description of each point of interest.

3 Semi-supervised Classification

Traditional supervised classification techniques use only labelled instances in the learning phase. In situations where the number of availabe labelled instances is insufficient, labelling is expensive and time consuming, semi-supervised classification can be employed, which uses both labelled and unlabelled instances for learning.

In the reported research, we used the following two methods for semisupervised classification.

3.1 Semisupervised Classification with Cluster Regularization

The principle of this method, in detail described in [8], consists in clustering all labelled and unlabelled instances

and estimating, for the instance x_k, $k = 1,\ldots,N$, its probability distribution q_k on the set of clusters. In addition, the following penalty function is proposed for the differences between the pairs (q_k, q_n) of probability distributions of the instances.

$$P(q_k,q_n) = \sin\left(\frac{\pi}{2}(r(q_k,q_n) * s(q_k,q_n))^{\kappa}\right), \quad k,n = 1,\ldots,N, k \neq n, \quad (1)$$

where $r(q_k,q_n)$ denotes the Pearson correlation coefficient between q_k and q_n, κ is a parameter controlling the steepeness of the mapping from similarity to penalty, and $s(q_k,q_n)$ is a normalized similarity of the probability distributions q_k and q_n, defined

$$s(q_k,q_n) = 1 - \frac{\|q_k - q_n\| - d_{\min}}{d_{\max} - d_{\min}} \quad (2)$$

using the notation

$$d_{\min} = \min Q,\ d_{\max} = \max Q, \quad \text{with } Q = \{\|q_k - q_n\| | k,n = 1,\ldots,N, k \neq n\}. \quad (3)$$

The results of clustering allow to assign pseudolabels to unlabelled instances. In particular, the pseudolabel assigned for the j-th among the M considered clusters to an unlabelled instance x_n in a cluster Ψ is

$$\hat{y}_{n,j} = \frac{\exp\left(\sum_{x_k \in \Psi \text{ is labelled}} y_{k,j}\right)}{\sum_{i=1}^{M} \exp\left(\sum_{x_k \in \Psi \text{ is labelled}} y_{k,i}\right)}, \quad (4)$$

where $y_{k,i}, i = 1,\ldots,M$ is a crisp or fuzzy label of the labelled instance x_k for the class i. For uniformity of notation, the symbol $\hat{y}_{k,j}, j = 1,\ldots,M$ can also be used for $y_{k,j}$ if x_k is labelled.

The penalty function (1) can be used as a regularization modifier in some loss function $L : [0,1]^2 \to [0,+\infty)$ measuring the discrepancy between the classifier outputs $F(x_n) = ((F(x_n))_1, \ldots, (F(x_n))_M)$ for an instance x_n, and the corresponding labels $(y_{n,1},\ldots,y_{n,M})$ or pseudolabels $(\hat{y}_{n,1},\ldots,\hat{y}_{n,M})$:

$$E = \frac{1}{N}\sum_{j=1}^{M}\left(\sum_{x_n \text{ labelled}} L((F(x_n))_j, y_{n,j}) + \sum_{x_n \text{ unlabelled}} \frac{\lambda \max(q_n)}{|\phi(x_n)|} \sum_{x_k \in \phi(x_n)} P(q_k,q_n) L((F(x_k))_j, \hat{y}_{k,j})\right), \quad (5)$$

where $\lambda > 0$ is a given parameter determining the tradeoff between supervised loss and unsupervised regularization, and the set of instances $x_k \neq x_n$ with the highest value of $P(q_k,q_n)$ is denoted $\phi(x_n)$.

In [8], the following design decisions have been made for the loss function and the classifier in (5):

1. The employed loss function can be derived from $D_{\mathrm{KL}}((\hat{y}_{n,1},\ldots,\hat{y}_{n,M}) \| F(x_n))$, the Kullback-Leibler divergence, from classifier outputs to labels or pseudolabels. If both the labels or pseudolabels and the classifier outputs form probability distributions on classes, then

$$D_{\mathrm{KL}}((\hat{y}_{n,1},\ldots,\hat{y}_{n,M}) \| F(x_n)) = = \sum_{j=1}^{M} \hat{y}_{n,j} \ln\left(\frac{(F(x_n))_j}{\hat{y}_{n,j}}\right), n = 1,\ldots,N. \quad (6)$$

Therefore, the considered loss function is

$$L((F(x_k))_j, \hat{y}_{k,j}) = = \hat{y}_{n,j} \ln\left(\frac{(F(x_n))_j}{\hat{y}_{n,j}}\right), n = 1,\ldots,N, j = 1,\ldots,M. \quad (7)$$

2. As a classifier, a multilayer perceptron with one hidden layer is used, such that the activation function g in its hidden layer is smooth and includes no bias, and its output layer performs the softmax normalization of the hidden layer. Hence,

$$(F(x))_j = \frac{\exp(g(w_{j\cdot}^{\top} x))}{\sum_{i=1}^{M} \exp(g(w_{i\cdot}^{\top} x))}. \quad (8)$$

The weight vectors $w_{1\cdot},\ldots,w_{M\cdot}$ in (8) are learned through the minimization of the error function (5).

3.2 Semi-supervised Kernel-Based Fuzzy C-means

This method, in detail described in [9], originated from the fuzzy c-means clustering algorithm [2]. Similarly to the original fuzzy c-means, the method is parametrized by a parameter $m > 1$. What makes this method more general than the original fuzzy c-means, is its dependence on the choice of some kernel K, i.e., a symmetric function on pairs (x,y) of clustered vectors, which has positive semidefinite Gramm matrices (e.g., Gaussian or polynomial kernels). In fact, the fuzzy c-means algorithm corresponds to the choice $K(x,y) = x^{\top} y$.

First, the membership matrix U^l is constructed, for clustering n_l labelled instances $x_1^l,\ldots,x_{n_l}^l$ into as many clusters as there are classes, i.e., M. For $j = 1,\ldots,M, k = 1,\ldots,n_k$,

$$U_{j,k}^l = \begin{cases} 1 & \text{if the instance } x_k^l \text{ is labelled with the class } j \\ 0 & \text{else.} \end{cases} \quad (9)$$

From U^l, the initial cluster centers are constructed as

$$v_j^0 = \frac{\sum_{k=1}^{n_l} U_{j,k}^l x_k^l}{\sum_{k=1}^{n_l} U_{j,k}^l}, j = 1,\ldots,M. \quad (10)$$

If for some $t = 0,1,\ldots$, the cluster centers $v_1^t,\ldots,v_M^t$ are available, such as (10), then they are used together with the chosen kernel K to construct the membership matrix

$U^{u,t}$ for clustering n_u unlabelled instances $x^u_1,\dots,x^u_{n_u}$, as follows:

$$U^{u,t}_{j,k} = \frac{(1-K(x^u_k,v_j))^{-\frac{1}{m-1}}}{\sum_{i=1}^{M}(1-K(x^u_k,v_i))^{-\frac{1}{m-1}}}, \quad j=1,\dots,M,\ k=1,\dots,n_u. \tag{11}$$

Finally, the cluster centers are updated, for $t=0,1,..$ by calculating

$$v_j^{t+1} = \frac{\sum_{k=1}^{n_l}(U^l_{j,k})^m K(x^l_k,v^t_j)x^l_k + \sum_{k=1}^{n_l}(U^{u,t}_{j,k})^m K(x^u_k,v^t_j)x^u_k}{\sum_{k=1}^{n_l}(U^l_{j,k})^m K(x^l_k,v^t_j) + \sum_{k=1}^{n_l}(U^{u,t}_{j,k})^m K(x^u_k,v^t_j)}. \tag{12}$$

The computations (11)–(12) are iterated until at least one of the following termination criteria is reached:

(i) $\|U^{u,t}-U^{u,t-1}\|<\varepsilon, t\geq 1$, for a given matrix norm $\|\cdot\|$ and a given $\varepsilon>0$;

(ii) a given maximal number of iterations $t_{\max}$.

4 Proposed Approach

4.1 Overall Strategy

Our methodology for the segmentation of video frames into foreground objects and background relies on the assumption that the user typically assigns corresponding labels to points of interest only in the first frame, and even not necessarily to all detected points of interest.

No matter whether the considered method of semisupervised classification is semisupervised classification with cluster regularization or semi-supervised kernel-based fuzzy c-means, the methodology always proceeds in the following steps:

1. In the first frame, the user labels some of the points of interest detected by the ORB detector.
2. Using the considered method of semisupervised classification, the remaining detected points of interest are labelled.
3. Matching points detected in the next frame are assigned the same labels as the points to which they are matched.
4. Using the considered method of semisupervised classification, the remaining points of interest detected in the next frame are labelled.
5. Steps 3 and 4 are repeated till either the points of interest in all frames have been classified or the scene has been so much disrupted between two frames that no points of interest could be matched between them (in such a case, new labelling by the user is needed).

4.2 Implementation of Object Segmentation

The Cartesian coordinates $([p]_1,[p]_2)$ of a point p of interest are expressed with respect to top left corner of the frame, using as units the frame height and width. Due to that, $[p]_1$ and $[p]_2$ are normalized to $[0,1]$.

For a match between points of interest p_k and p_{k+1} in subsequent frames k and $k+1$, the following criteria have been used:

(i) The point p_{k+1} must lie within the radius r^p_k from the estimated new position of the point $\hat{p}_k$

$$\|p_{k+1}-\hat{p}_k\|<r^p_k. \tag{13}$$

Here, the estimated position $\hat{p}_k$ is calculated as

$$\hat{p}_k = \begin{cases} p_k + c_1(p_k-p_{k-1}) & \text{if } p_{k-1} \text{ is available,} \\ p_k & \text{else,} \end{cases} \tag{14}$$

where $c_1>0$, and the radius r^p_k is calculated as

$$r^p_k = (u^p_k W)^2, \tag{15}$$

where u^p_k quantifies the uncertainty pertaining to the point p_k in the k-th frame and W denotes the frame width (in the units in which point positions are expressed). The uncertainty u^p is set to $u^p_1=c_2>0$ in the first frame and is then evolved from frame to frame through linear scaling above a lower limit $c_3>0$:

$$u^p_{k+1} = \begin{cases} \max(c_3, c_4 u^p_k) & \text{if } p_k \text{ is matched,} \\ c_5 u^p_k & \text{if } p_k \text{ is not matched,} \end{cases} \tag{16}$$

where $0<c_4<1, c_5>1$.

Moreover, if the evolution (16) leads to $u^p_{k+1}>c_6$ for some $c_6>c_3$, then the point p is deactivated and not any more considered for matching.

(ii) Hamming distance between the 256-bit binary desciptors of the points is at most 64.

The choice of the real-valued constants in the criterion (i) has been based on the resolution of the video (4K), on the frame rate (25) and on the defaults in the ORB implementation based on [7]. They have been set to the following values: $c_1=0.6, c_2=0.02, c_3=0.009, c_4=0.9, c_5=1.1, c_6=0.03$.

In each frame, the described implementation was used to find 500 most interesting points. On a linux computer with a 3.3 GHz Intel Xeon E3-1230 processor, this took 95.32 ms.

4.3 Implementation of Semi-supervised Classifiers

As input features for both classification methods, the Cartesian coordinates $([p_k]_1,[p_k]_2)$ of the point in the k-th frame and and the polar coordinates $([p_k-p_{k-1}]_{||},[p_{k+1}-$

$p_k]_\varphi$) of its movement with respect to the previous frame are used.

In the implementation of the semisupervised classification with cluster regularization method described in 3.1, we used k-means clustering for an initial clustering of all instances. Although this method allows choosing the number of clusters independently of the number of classes, we have set it to the same value for comparability with semi-supervised kernel-based fuzzy c-means, i.e., to the value 2 corresponding to the classes of foreground objects and background. Hence, we performed k-means clustering with $k = 2$. Since the k-means algorithm does not output a probability distribution on the set of clusters, we employed a simple procedure proposed in [8] to transform the original distances from an instance x_n to cluster centers $v_1, \ldots, v_k$, to a probability distribution q_n, which assures that x_n more likely belongs to clusters to which centers it is closer:

$$(q_n)_i = \frac{1 - \left(\frac{\|x_n - v_i\|}{\sum_{j=1}^{k} \|x_n - v_i\|} \right)}{k-1}. \tag{17}$$

Consequently, for our case $k = 2$:

$$(q_n)_1 = \frac{\|x_n - v_2\|}{\|x_n - v_1\| + \|x_n - v_2\|}, \tag{18}$$

$$(q_n)_2 = \frac{\|x_n - v_1\|}{\|x_n - v_1\| + \|x_n - v_2\|}. \tag{19}$$

The remaining parameters pertaining to semisupervised classification with cluster regularization were set as proposed in [8]: $\lambda = 0.2, \kappa = 2, |\phi(x_n)| = 10$.

For the semi-supervised kernel-based fuzzy c-means algorithm described in 3.2, we used a Gaussian kernel function for updating the membership matrix $K(x,y) = exp(-\|x-y\|^2/\sigma^2)$, where the parameter σ is computed as proposed in [9]:

$$\sigma = \frac{1}{M} \sqrt{\frac{\sum_{n=1}^{N} \|x_n - v\|^2}{N}}, \tag{20}$$

where v is the center of all instances. The remaining parameters were set as follows: $m = 2, \varepsilon = 0.001, t_{max} = 50$.

5 Experimental Validation

5.1 Employed Data

For the validation of the proposed approach we prepared 12 short videos. In all videos, there is a yellow or blue balloon as a foreground object and a green background. On the background, there are a few small red sticky notes to help detecting some key points. The videos were recorded in a UHD resolution.

Here is a brief characterization of all employed videos:

- a handheld camera, both the foreground object and the background are sharp,
- a handheld camera, only the foreground object is sharp (2 videos),
- a static camera, only the background is sharp (2 videos),
- a static camera, only the background is sharp, the foreground object is close to the camera,
- a static camera, only the foreground object is sharp, a hand is interfering with the background (2 videos),
- a static camera, only the foreground object is sharp, it is moving towards the camera,
- a static camera, only the foreground object is sharp, it is moving away from the camera,
- static camera, only the foreground object is sharp, it passes the scene multiple times (2 videos).

For the testing, labels were available for all points of interest. Unfortunately, those labels were often unreliable.

5.2 Results and Their Analysis

On all the employed videos, we measured the quality of classification by means of accuracy, sensitivity, specificity and F-measure of both implemented classification methods.

For the fuzzy c-means method, the accuracy and specificity on the unlabelled data are illustrated for four particular videos in Figure 1.

For the cluster-regularization method, we compared the values of the considered four quality meaures obtained with five classifiers trained in each of the five first video frames with respect to the delay between classifier training and measuring its quality. The results of their comparison are for three particular delays, 1 frame, 5 frames and 10 frames, summarized in Table 1. In addition, for delays up to 50 frames, they are again illustrated for accuracy and sensitivity on the four videos used already in connection with the fuzzy c-means classifier, in Figures 2–5.

The figures (2)–(5) indicate that classifiers trained in a later frame tend to have higher accuracy and specificity, but in general, the differences between classifiers trained in different frames are small. This is confirmed by the Friedman test for delays 1, 5 and 10 frames between classifier training and measuring its quality and for all four considered quality measures. The hypothesis of equality of all five classifiers is rejected (p-value $< 5\%$) only for the delay 1 frame and the F-measure, and weakly rejected (p-value $< 10\%$) for the delay 1 frame and the sensitivity, as well as for the delay 5 frames and the F-measure. A posthoc test expectedly reveals that the equality of all five classifiers was rejected mainly due to differences between classifiers trained in the early and in later frames; in particular between those trained in the 1st and 4th frame (delay 1, both sensitivity and F-measure), classifiers trained

Table 1: Comparison of the values of the considered quality measures obtained with classifiers trained in each of the 5 first video frames for different delays between classifier training and testing, obtained on data from the 12 employed videos. The result in a cell of the table indicates on how many videos the considered measure of classifier quality (accuracy, sensitivity, specificity, F-measure) was higher for the row classifier : on how many videos it was higher for the column classifier. A result in italic, respectively bold italic, indicates that after the Friedman test at least weakly rejected (p-value $< 10\%$) the hypothesis that the considered quality measure is equal for all classifiers (cf. Table 2), the post-hoc test according to [3, 4] weakly rejects, respectively rejects (p-value $< 5\%$) the hypothesis that it is equal for the particular row and column classifiers. All simultanously tested hypotheses were corrected in accordance with Holm [5]

	Delay between the frame on which the classifier is trained and the frame when it is tested														
	1 frame					5 frames					10 frames				
# 1	Frame in which the compared classifier was trained # 2														
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
	Accuracy														
1		5:7	5:7	7:5	6:6		4:8	7:5	7:5	10:2		4:8	4:8	4:8	2:10
2	7:5		4:8	8:4	6:6	8:4		6:6	7:5	10:2	8:4		7:5	5:7	5:7
3	7:5	8:4		9:3	6:6	5:7	6:6		7:5	11:1	8:4	5:7		6:6	5:7
4	5:7	4:8	3:9		5:7	5:7	5:7	5:7		10:2	8:4	7:5	6:6		5:7
5	6:6	6:6	6:6	7:5		2:10	2:10	1:11	2:10		10:2	7:5	7:5	7:5	
	Sensitivity														
1		8:4	8.5:3.5	*9.5:2.5*	8.5:3.5		8:4	7:5	7:5	9:3		6.5:5.5	8:4	8.5:3.5	8.5:3.5
2	4:8		8:4	10:2	9:3	4:8		6:6	7.5:4.5	9.5:2.5	5.5:6.5		6:6	8:4	8:4
3	3.5:8.5	4:8		10.5:1.5	9.5:2.5	5:7	6:6		7.5:4.5	8.5:3.5	4:8	6:6		8.5:3.5	9.5:2.5
4	*2.5:9.5*	2:10	1.5:10.5		6.5:5.5	5:7	4.5:7.5	4.5:7.5		8:4	3.5:8.5	4:8	3.5:8.5		8.5:3.5
5	3.5:8.5	3:9	2.5:9.5	5.5:6.5		3:9	2.5:9.5	3.5:8.5	4:8		3.5:8.5	4:8	2.5:9.5	3.5:8.5	
	Specificity														
1		7.5:4.5	6.5:5.5	7:5	7:5		3.5:8.5	5:7	5:7	4:8		6.5:5.5	3.5:8.5	2:10	3.5:8.5
2	4.5:7.5		4.5:7.5	6:6	6.5:5.5	8.5:3.5		5:7	4.5:7.5	4:8	5.5:6.5		4:8	4:8	4:8
3	5.5:6.5	7.5:4.5		6:6	7:5	7:5	7:5		7:5	6.5:5.5	8.5:3.5	8:4		4.5:7.5	6:6
4	5:7	6:6	6:6		4.5:7.5	7:5	7.5:4.5	5:7		4:8	10:2	8:4	7.5:4.5		6.5:5.5
5	5:7	5.5:6.5	5:7	7.5:4.5		8:4	8:4	5.5:6.5	8:4		8.5:3.5	8:4	6:6	5.5:6.5	
	F-measure														
1		8:4	9:3	***10:2***	8:4		6:6	7:5	8:4	***11:1***		5.5:6.5	9:3	8.5:3.5	9.5:2.5
2	4:8		7:5	***12:0***	9:3	6:6		6.5:5.5	7:5	*10:2*	6.5:5.5		6.5:5.5	7.5:4.5	9.5:2.5
3	3:9	5:7		11:1	8:4	5:7	5.5:6.5		8:4	*11:1*	3:9	5.5:6.5		8:4	9:3
4	***2:10***	***0:12***	1:11		6:6	4:8	5:7	4:8		8.5:3.5	3.5:8.5	4.5:7.5	4:8		9:3
5	4:8	3:9	4:8	6:6		***1:11***	*2:10*	*1:11*	3.5:8.5		2.5:9.5	2.5:9.5	3:9	3:9	

Table 2: Results of the Friedman test of the hypothesis that for a given delay between classifier training and measuring its quality, a given quality measure is equal for the classifiers trained in each of the 5 first video frames, for the 12 combinations of delays and quality measures considered in Table 1. The combinations for which the tested hypotheseis was weakly rejected (p-value $< 10\%$) are in italic, the single combination for which it was rejected (p-value $< 5\%$) is in bold italic. All simultanously tested hypotheses were corrected in accordance with Holm [5]

Quality measure	Delay	p-Value
accuracy	1	1
accuracy	5	0.117
accuracy	10	1
sensitivity	1	*0.052*
sensitivity	5	0.428
sensitivity	10	0.238
specificity	1	1
specificity	5	1
specificity	10	0.25
F-measure	1	***0.043***
F-measure	5	*0.089*
F-measure	10	0.238

in the 1st and 4th frame (delay 1, F-measure) and classifiers trained in the 1-3 frame and in the 5th frame (delay 5, F-measure).

6 Conclusion

The presented research integrates two comparatively recent approaches, the keypoint detector ORB, which is a combination of a corner detection method with a visual descriptor method, and two semi-supervised classifiction methods. To our knowledge, this is the first time these approaches are used together for the task of scene segmentation into the foreground objects and the background.

On the other hand, this is a work in progress and the presented results are still rather preliminary, being obtained on 12 artificially created videos with a quite simple scene segmentation. Both approaches should be investigated in the context of more complex segmentations and more realistic scenes. To this end, however, especially the ORB detector needs to be more deeply elaborated with methods of semisupervised classification.

Acknowledgement

The research reported in this paper has been supported by the Czech Science Foundation (GAČR) grant 18-18080S.

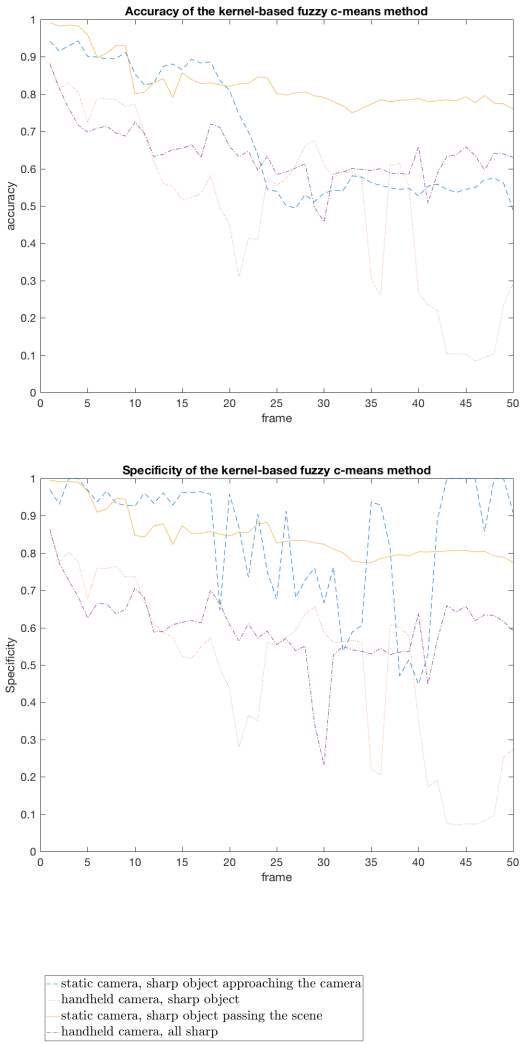


Figure 1: The evolution of accuracy (top) and specificity (bottom) of the c-means method on the unlabelled data for four particular videos

References

[1] M.S. Allili, N. Bouguila, and D. Ziou. Finite general Gaussian mixture modeling and application to image and video foreground segmentation. *Journal of Electronic Imaging*, 17:paper 013005, 2008.

[2] J.C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York, 1981.

[3] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[4] S. Garcia and F. Herrera. An extension on "Statistical Comparisons of Classifiers over Multiple Data Sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694, 2008.

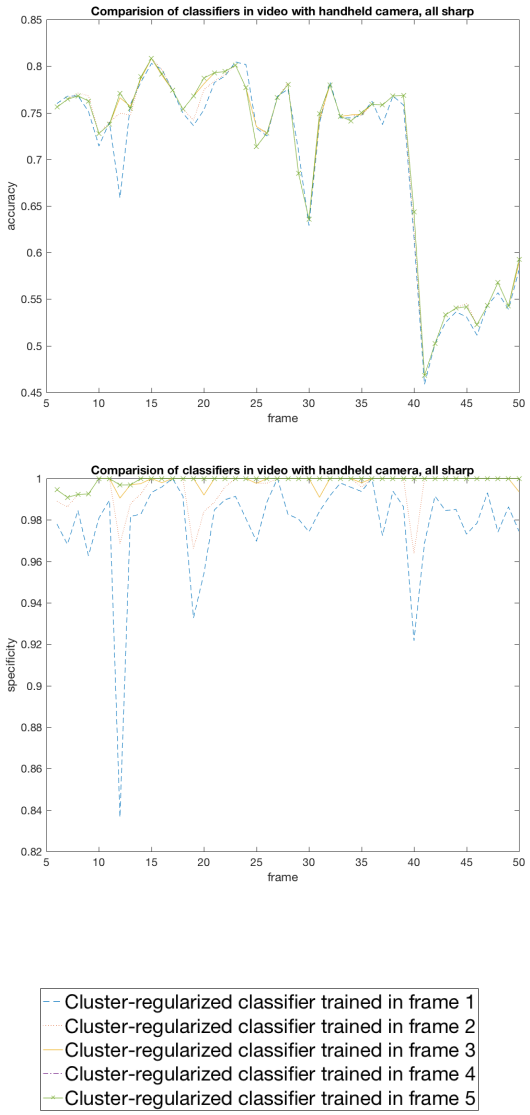


Figure 2: The evolution of accuracy (top) and specificity (bottom) of the classifiers trained in each of the 5 first video frames for a handheld-camera video with both the foreground object and the background sharp

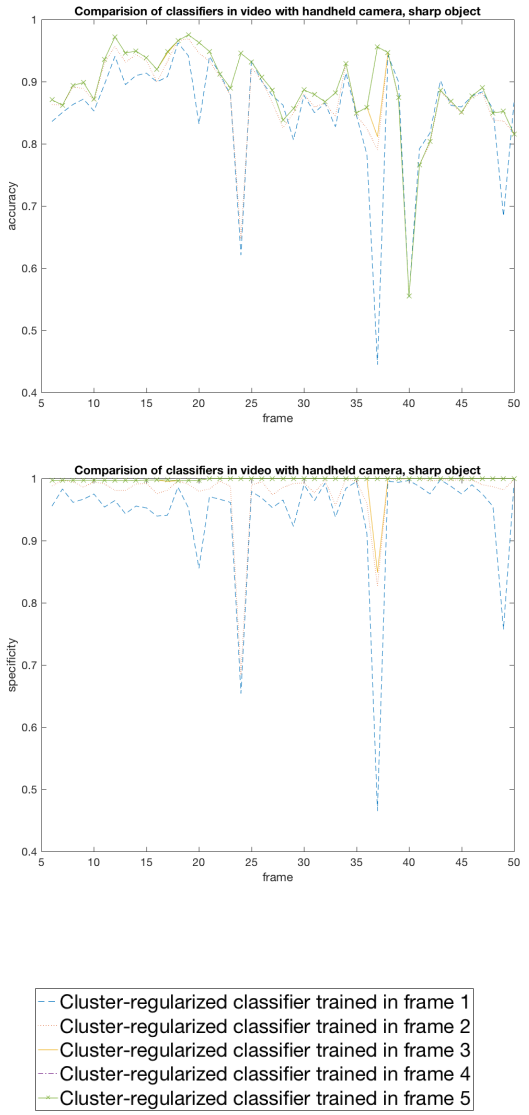


Figure 3: The evolution of accuracy (top) and specificity (bottom) of the classifiers trained in each of the 5 first video frames for a handheld-camera video with only the foreground object sharp

[5] S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6:65–70, 1979.

[6] L. Li, W. Huang, I.Y.H. Gu, and Q. Tan. Foreground object detection from videos containing complex background. In *11th ACM Conference on Multimedia*, pages 2–10, 2003.

[7] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *International Conference on Computer Vision*, pages 2564–2571, 2011.

[8] R.G.F. Soares, H. Chen, and X. Yao. Semisupervised classification with cluster regularization. *IEEE Transactions on Neural Networks and Learning Systems*, 23:1779–1792, 2012.

[9] D. Zhang, K. Tan, and S. Chen. Semi-supervised kernel-based fuzzy c-means. In *ICONIP'04*, pages 1229–1234. Springer, 2004.

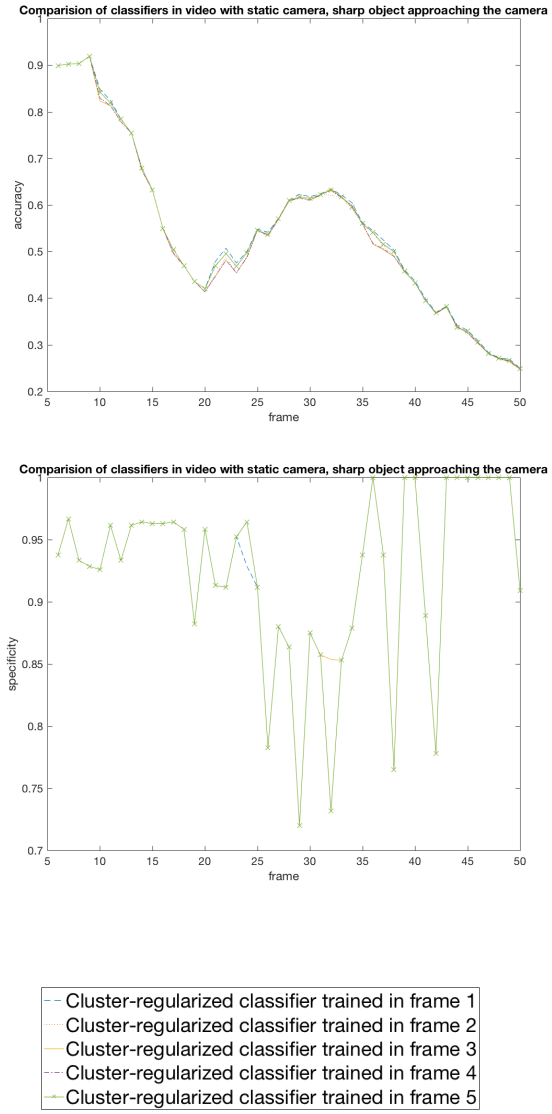


Figure 4: The evolution of accuracy (top) and specificity (bottom) of the classifiers trained in each of the 5 first video frames for a static-camera video, in which only the foreground object is sharp and is moving towards the camera

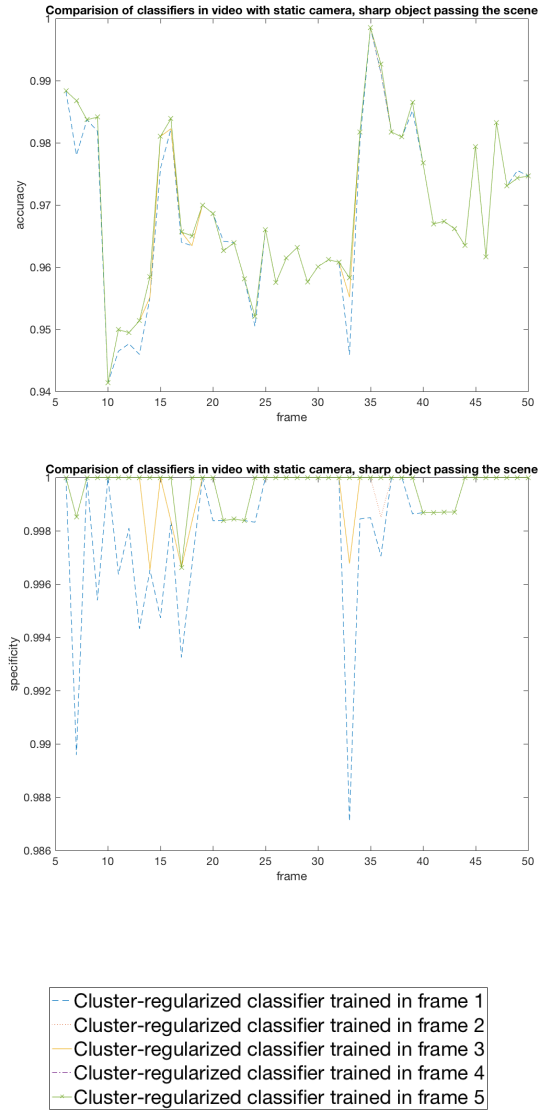


Figure 5: The evolution of accuracy (top) and specificity (bottom) of the classifiers trained in each of the 5 first video frames for a static-camera video, in which only the foreground object is sharp and passes the scene multiple time

S. Krajči (ed.): ITAT 2018 Proceedings, pp. 108–115
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Andrea Galloni, Balázs Horváth, and Tomáš Horváth

Real-time Monitoring of Hungarian Highway Traffic from Cell Phone Network Data

Andrea Galloni, Balázs Horváth, and Tomáš Horváth

Department of Data Science and Data Technologies,
Faculty of Informatics,
ELTE – Eötvös Loránd University in Budapest
{andrea.galloni,balazs.horvath,tomas.horvath}@inf.elte.hu,
http://t-labs.elte.hu/

Abstract: A lightweight model for real-time monitoring of the load of Hungarian highway traffic is presented in the paper. The input data of the model are cell phone network event records provided by Magyar Telekom Nyrt., the major Hungarian telecommunication company. The output is a classification of the level of crowdedness of the Hungarian highways inferred from the activity level of the mobile telecommunication infrastructure. While processing, a data-stream is flowing through a chain of simple but efficient data structures. For computing anomalies against the usual behavior of the traffic at given segments of the highway, so-called break-points, known from the SAX representation of time-series, are utilized which require cheap computation. The model is implemented as a server application able to feed a client web-based visualization application implemented for demonstration purposes. The experiments, performed on anonymized data covering one month of cell phone records, show that the presented model is computationally cheap, it efficiently runs even on low-end hardware such that Raspberry Pi.

Keywords: Anomaly Detection, Mobile Data Analytics, Visualization

1 Introduction

A method resulting from an industrial research project is presented in this paper. The presented method has been developed for a specific and well-defined use case: inferring the level of the mobility traffic load on Hungarian highways from cell phone network data. The data have been provided by the industrial partner, *Magyar Telekom Nyrt.*, the major Hungarian telecommunication company, a subsidiary of *Deutsche Telekom AG*.

Experimental outcomes are positive and promising as the underlying core model is computationally light and simple. The data structures used and the overall system architectural-design could be, possibly, exploited for other applications or use cases. More specifically, the presented model can be used where there is the need to detect anomalies in time series given a set of nodes and the logs providing quantitative information describing the activity of such nodes over time. Even if the developed framework is specifically build to infer information regarding the Hungarian highways mobility infrastructure through the analysis of the mobile telecommunication infrastructure, the core model can be adapted to different scenarios and different data logs such as tower cell crowdedness or Internet backbones nodes activity monitoring.

1.1 Related Work

Nowadays we are witnessing to a constant increasing speed of networks, furthermore the capability to store and process conspicuous amount of data can be performed at affordable prices. This new scenario enables telecom operators to store and process big quantities of logs triggered by a countless number of events. Along the past years, the research community proposed several models regarding the possibility to infer or predict information regarding the status of the mobility infrastructure analyzing the mobile telecommunication event logs. Furthermore, the evolution of new telecommunication technology standards such as 5G will bring more efficient and accurate localization techniques leading to more precise analysis and estimations [7].

In [6], authors present quite a complex model able to estimate the traffic flow making use of anonymized temporal series of cell handover logs, building state diagrams and using Markov Models in order to detect car accidents.

On the other hand, in [8], authors propose a framework mining several heterogeneous data sources making use of the MapReduce programming-model (more precisely using Apache Hadoop) in order to process big amounts of data in a reasonable amount of time involving high-end hardware and clusters of machines. The authors of this contribution are able to estimate the traffic volume and the speed of the traffic flow.

In [9], the authors provide a full overview of methodologies providing a possible list of necessary steps in order to infer traffic information such that i) location data collection, ii) terminal classification in order to determine which mobile terminals are located on the road and which means of transport they are in, while iii) map matching phase in order to link the extracted location data with the mobility infrastructure, iv) the route determination process used to determine the path of the vehicles while the last step is to perform v) the estimation of the traffic state.

In order to estimate traffic flows the use of origin/destination matrices have been tried out in [10] and [11],

however, as pointed out in [6], these solutions might be computationally expensive from the technical point of view and hard to scale when the size of the user set tends to grow. On the other hand, from the law regulation perspective these solutions might see limitations on real deployment scenarios due to privacy concerns and strict regulations, especially the ones applying within the European Union.

Within this contribution, a minimalistic approach to traffic load detection is introduced exploiting just events triggered by the active utilization of the *User Equipments* (Calls, SMS and Mobile Data Usage) without involving logs related to lower level signalling protocols such as cell handover event logs. The aim of this research was to discover at which extent and precision is possible to infer reliable traffic analysis with minimalistic datasets and minimal computing costs. A server application able to feed a web-based visualization application has been implemented for demonstration purposes. Experiments provided on real but anonymized data covering one month of cell phone records show that the the presented model is promising and is able to efficiently run even on low-end hardware.

The rest of this paper is organized as follows: Section 2 gives a description of the available data and the procedure utilized to match telecom data with geographical-map data. In Section 3, a detailed description of the system architecture and the core estimation model are provided. In the Section 4 the process of traffic load classification is described. The following Section 5 contains experimental results and measurements. Finally, in Section 6 some conclusions and plans for future work are provided.

2 Data Sources and Framework Initialization

An important part of the presented framework is its initialization to the specific use case, such that highway traffic load monitoring, in our case, what consists of understanding the data and selecting the towers to be considered relevant by the framework.

2.1 Telecom Data

The industrial partner has granted access to a dataset[1] containing several `.csv` (Comma Separated Values) files organized in a daily basis containing two main kinds of information such that

Call Detail Record (CDR) data concern the activity logs of each user interacting with the network on a daily basis. A CDR is produced by a telephone equipment that documents the details of a call or other telecommunications events (e.g. notifications, short message service or signaling protocols) that involves the telecom provider infrastructure.

The dataset is composed by several files accounting a size of 500GB. The entire information contained within the dataset covers a range of 31 days, more precisely between 15th September 2016 and 15th October 2016, where all the unique identifiers referencing to the users have been anonymized on a daily basis. The overall number of logs is around 200 million records per day.

In order to work with just the useful data all the unnecessary information contained in the dataset such as the nature of event logs and other information regarding customer related data (e.g. phone and events identifiers) have to be discarded by the framework.

At the end of this process, the CDR files contains three kind of attributes as presented in Table 1, namely, the *Unique User Identifier* (UUID) re-anonymized on a daily basis (to prevent tracking of user movement across more days), the *date-time* information related to the log event and the *Tower Identifier* (TID) providing information from which cell-tower the event has been triggered.

Cell Reference (CR) data provide informations about the mobile radio-towers and their positioning within the Hungarian territory. As illustrated in the Table 2, CR data contain three attributes, namely, the *Tower Identifier* (TID) which connects the CDR data with the CR data, the *Latitude* and the *Longitude* regarding the given tower. Due to how the industrial partner gathered and anonymized the data, process on which the authors were not involved, some records contained within the CDR files hold UUIDs with *NULL* value or in some cases hold inconsistent TIDs. Namely some TIDs contained in the CDR logs do not match any of the TIDs in the CR data. The UIIDs inconsistencies are uniformly spread over the locations while for the inconsistent TIDs is not possible to draw any conclusion about the geographical regions affected. In case of those inconsistent logs, it is not possible to get the subject performing the action or the location of the event. For this reason all the affected records, which are around 30% of all the records, are affected and have to be handled (discarded) by the framework during the on-line computation.

2.2 Geographic Maps Data

In order to get information about Hungarian highways the research relied on OpenStreetMap[2] (OSM) data. The open project makes available data about roads, trails, cafes, railway stations and other basic map features from all around the world.

[1] Due to the signed non-disclosure agreement between the academic and the industrial partner of the project, each sample of the data, e.g. the Tables 1 and 2, presented in this paper are synthetic, i.e. contain fictive information.

[2] `https://www.openstreetmap.org/`

UUID	date-time	TID
6776554S3449	2016-09-15 13:30:41	10B32F03E10CB
777865354435	2016-09-15 00:27:50	10DA0232324BC
677655453449	2016-09-15 17:44:08	00443344DFFEA

Table 1: A synthetic CDR data sample with fictive UUID, date-time and TID data, for illustration purpose .

TID	Latitude	Longitude
6776554S3449	46.291311	17.366325
777865354435	46.282342	17.357244
677655453449	46.366745	17.364112

Table 2: A synthetic CR data sample with fictive TID, Latitude and Longitude data for illustration purpose.

First of all information about the Hungarian country borders have been extracted, then the data regarding only highways has been kept obtaining a `.json` file containing informations about each highway divided in several segments, where each segment contains information describing a small section of the highway (e.g.: name, type, speed limit) and its location. The length of each highway's section depends on the topography of the area, the density of the population and the radio-technology of the cell towers of the operator. The outcome of this phase, i.e. the detected borders, can be observed in the Figure 6.

Detecting Relevant Towers In order to monitor the highways infrastructure traffic and exclude irrelevant information from the data model an additional filtering and selecting step has been performed. After this phase, only the relevant towers that have a strict correlation with the highways infrastructure have been kept.

The density of the cell tower placement and its spatial characteristics represent a crucial issue in terms of space-resolution within the developed monitoring system. Provided the mostly flat characteristics of the Hungarian landscape, is possible to assume that the cell towers displacement is mostly not conditioned by the topographic properties of the surrounding areas but rather follows the density of the population over the whole territory. This characteristic of the cell towers placement is due to several factors such as scalability, laws of physics and signal processing theory. Figure 1 illustrates the density of the city of Budapest and its east country side area from which it is possible to observe that in rural areas cell towers are placed close to the highway in order to provide the radio-signal to travelers.

In order to detect cell towers which lead to the crowdedness status of that specific section of the highway, an ad-hoc algorithm have been developed based on a *QuadTree* [1] data structure. The generated tree contains all the coordinates of the cell towers that are listed in the CR data. Then, for every highway section the closest tower cell has been found querying the tree structure. After this process,

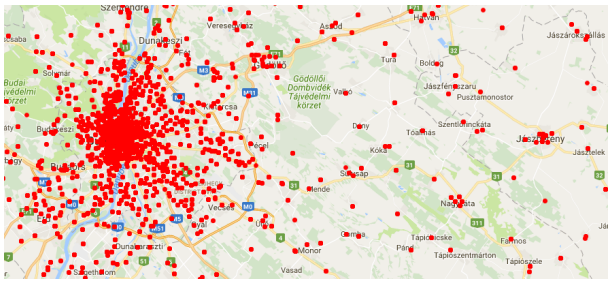


Figure 1: Density of cell towers for Budapest Urban and Eastern Rural Area, an illustrative example.

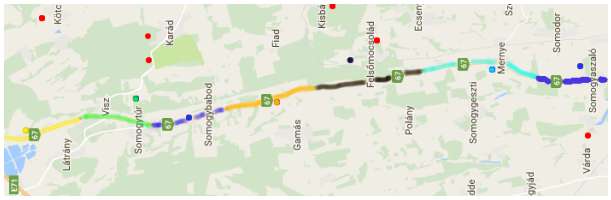


Figure 2: Relevant towers identification and towers competence attribution, an illustrative example.

as shown in Figure 2, each highway section is linked to a specific cell tower while all the towers not related with the highways will not be considered by the framework while processing.

3 The System Architecture

The framework has its roots in several components, i.e. the following data structures (called dictionaries, according to Python notation) and logical units.

3.1 Data Structures

Relevant Towers Dictionary (RTD) RTD is one of the main data structure on which most of the others are based on. In fact the RTD represents an `HashMap` having as keys the identifiers (TID, see Table 2) of all the relevant towers and as values the geographical coordinates of given towers as strings. RTD is the result of the module for detection of relevant towers, described above. This dictionary remains constant in the framework and changes only if there are changes in geographical locations of cell towers such that a new tower is placed near the highways, for example.

User at Relevant Towers Dictionary (URTD) URTD is a `HashMap` and it has as keys all the UUIDs (Unique User IDs, see Table 1) of the users whose previous logs were triggered by relevant towers, namely, those logs who had their TID appearing in the RTD keys, and, as values the TID of the tower the given user has been related to for the last time. At the startup of the system, this data structure is empty and at the beginning of a new day, due to the UUID re-anonymization on a daily basis, it is reinitialized.

Tower ID Counter Dictionary (TIDCD) TIDCD is a `HashMap` and it has as keys (TIDs) all the relevant towers while it has as values counters representing the number of users whose last log have been related to that specific TID.

3.2 The Status Maintainer Module (SMM)

As soon as a log event is triggered, the SMM is delegated to keep track of the last location of the user (attaching to him/her the proper TID in the URTD) and increases or decreases the counter of the proper TID within the TIDCD according to the situation. This module acts as a supervisor moving the subscribers from a tower to another, keeping the model up to date with the information provided by the event logs. SMM is delegated to interpret and take decisions based on the information provided from the logs with the following functionality:

As soon as the application is started, both the URTD and the TIDCD Hash Maps are empty. As the first incoming log having a TID present in the RTD key-set is processed, the SMM will fill the URTD with the UUID as key and the TID as value, then, it will initialize the counter of the specific TID key within the TIDCD to 1. If a second log from the same user will come but, this time, with a different and relevant TID then the counter of the old TID (the last "location" of the user) within the TIDCD will be decreased by one unit and suddenly the corresponding URTD's value will be updated to the actual TID inferred from the log querying the RTD and, finally, the corresponding TIDCD value (for the actual TID the user is connected to) will be increased. In the last case, if the subscriber's handset will trigger a log which is not related to a relevant tower, the counter of the old TID within the TIDCD will be decreased by one and the key within the URTD corresponding to the specific UUID will be removed (the user has left the set of towers delegated to monitor the highways mobility status). All the event logs (records) containing a non-relevant TID (user is not on a highway) and UUID not stored in URTD (user was not on a highway before) are immediately discarded. Figure 3 provides an illustration of the SMM logic.

3.3 The Evaluator Module (EM)

The EM is the core module of the framework, which is responsible for classifying the status of the load of a specific highway segment. It uses as an evaluation model described in the next section where the number of classes is determined by a parameter.

The EM module is triggered every time a time frame (a parameter of the framework) expires. At this point it is time to evaluate the status of the whole system. At this stage the EM iterates over all the TIDCD, computes all the means and standard deviations using the past data necessary to evaluate the actual status of all the relevant TIDs and then finally classifies every segment of a given highway into one of the predefined classes. For example, in case of 10 classes, the class 5-6 corresponds to normal traffic, 7-8 to higher while 9-10 to very high traffic, 3-4 to lower and 1-2 to very low traffic on a given segment of the highway.

3.4 The Notification Delegate Module (NDM)

The NDM is the part of the framework responsible to constantly collect the result of the EM and update the clients about the highways status sending all the needed informations as a `json` payload. While this process takes place, an ID conversion is performed. In fact, the back-end and the front-end of the framework, due to security reasons, do not share the same internal IDs for representing the TIDs. Once finished, the control is passed to the Notification Delegate Module responsible for communicating with the client (described below).

4 Classification of the Traffic Load

In order to classify the load of a segment of highway, one should consider the past data as a reference for the evaluation of the new entries. Given the topographic features of the Hungarian landscape it is expected that the activity of cell towers close to the highway have a low *static noise* (in [6] authors highlights that systems relying on cell telecom logs for traffic estimations within urban areas could suffer high loss of precision due to event logs triggered by non-traveling users). In fact, in this specific case the majority of the event logs are generated mostly by travelers and it is easily possible to observe that the number of travelers in a given time-range tend to be similar for different week-days.

Provided the latter observation, it is possible to build a model such that in order to classify the load of a specific road segment in a precise time-range of the day (e.g.: between 12:00 and 12:15) compares the value to be classified against the values of load within the same time range of the previous days. Here, a sort of seasonality of the traffic has to be considered, e.g. there is more heavier load during the rush hours while less traffic at nights.

An other, interesting phenomenon is that sometimes a traffic anomaly can become normal with time. For example, consider a longer construction work on a highway causing the close of some parts (e.g. lanes) of the road. In

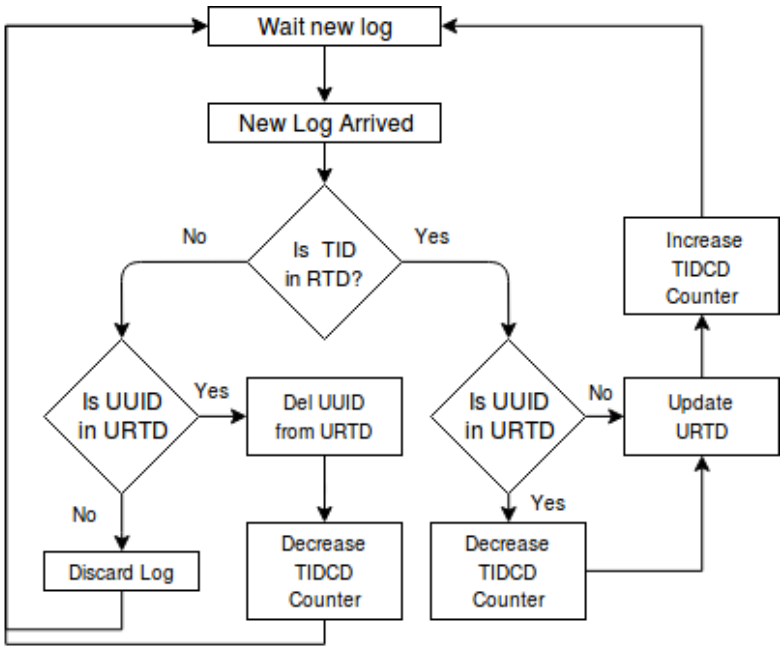


Figure 3: The logic diagram of the Status Maintainer Module

the time of its appearance, since it is sudden for the traffic, it is considered an anomaly and results in traffic jams. However, with time, the traffic normalizes such that people get used to it (e.g. start using alternative routes) and the notion of heavy load changes.

All of the above observations lead to the straightforward use of time-series for representing the given problem of traffic load monitoring and classification.

4.1 Utilizing Breakpoints

The proposed model for classification of traffic load in various highway segments utilizes well-known concepts in Symbolic Aggregate Approximation (SAX) of time series [2, 3]. SAX makes use of Piecewise Aggregate Approximation (PAA), a computationally very cheap method [4] since it operates with *arithmetic mean* and *standard deviation*, both very cheap operations.

SAX allows a time series of arbitrary length n to be reduced to a string of arbitrary length w, ($w \ll n$) with the alphabet size $a > 2$ (the number of letters used to represent the time-series using SAX). In this process the data is divided into w equal sized "frames". The mean value of each frame is calculated and a vector of these values becomes a reduced representation. For further details, refer to [2, 3].

Assuming that the distribution of the values over the time-series follows a normal distribution, it is possible to subdivide the time-series into so called *breakpoints B*. Breakpoints are a sorted list of numbers $B = (\beta_1, \cdots, \beta_{a-1})$ such that the area under a $N(0,1)$ Gaussian curve from β_1 to $\beta_{i+1} = 1/a$ where β_0 and β_a are defined as $-\infty$ and $+\infty$, respectively. The advantage of utilizing breakpoints is that they do not need further computation but may be determined by simply looking them up in a statistical table, as illustrated in the table 3 containing the breakpoints dividing a Gaussian distribution in an arbitrary number (from 3 to 10) of regions.

The final key-concept of the proposed model is that it does not represent the data in the past with a SAX representation, however the system classifies a new entry using the breakpoints generated making use of the data recorded in past in a SAX-like fashion just performing a lookup on a small table.

An efficient way to detect anomalies in time series is that it is enough to compare the breakpoint determined for the actual time frame to the breakpoints determined for the corresponding time frame(s) in the past, for example, the same time of the day before or the same time of the same day a week before, etc. Depending on the number of classes to which the framework should classify the new entry in the dataset, the right column of the statistical table has to be implemented in the system and then looked up. In order to give a wider freedom of tuning, this value has been kept as a parameter of the framework.

Historical Data Representation The data from the past are stored in an efficient-to-load binary format making use of Python's Pickle module. Files are stored in a daily format in the form of a `HashMap` of arrays having as keys the TIDs. Once loaded, the files are reshaped in *MxN* matrices, one for each TID, where M is the number of timeframes and N represents the number of days to consider in the past. All the tests in this research were performed setting the time frames at 15 minutes and considering the last

	a							
β_i	3	4	5	6	7	8	9	10
β_1	-0.43	-0.67	-0.84	-0.97	-1.07	-1.15	-1.22	-1.28
β_2	0.43	0	-0.25	-0.43	-0.57	-0.67	-0.76	-0.84
β_3		0.67	0.25	0	-0.18	-0.32	-0.43	-0.52
β_4			0.84	0.43	0.18	0	-0.14	-0.25
β_5				0.97	0.57	0.43	0.14	0
β_6					1.07	0.67	0.43	0.25
β_7						1.15	0.76	0.52
β_8							1.22	0.84
β_9								1.28

Table 3: Statistical table for determining the values of breakpoints

15 days in the past. This representation have been chosen because once the system is running in real-time with a continuous data-stream then it is easy to shift the matrix data and recompute all the means and standard deviations efficiently. This shifting and re-computation operation would be performed once per day at a given time (e.g. midnight) exactly when the system is subject of a reinitialization or when the log re-anonymization process is performed.

5 Experiments

Given the lack of open systems providing precise information about the traffic flows over time we validated the system over specific traffic congestions. The goal of the experiment is to validate the method looking for a correlation between the highways status and the telecommunication infrastructure. In order to validate the results of the developed system authors asked the collaboration of *utinform*[3], a department of *Magyar Közút Nonprofit Zrt.* whose responsibility is to collect information and monitoring the roads traffic. They gather information from heterogeneous data sources: the employees of the company monitoring the highways, the county directorates and the engineering departments employees. Utinform, in order to cross-validate the crowd sourced data sets, is also cooperating with institutions such as police, disaster management, and public transport which have their own feed to post their information to the system. Furthermore, the system has a crowd source interface, where people can submit experienced traffic anomalies. The goal of utinform is to provide fast and validated traffic information to the drivers in whole Hungary.

The data received include information regarding traffic congestions on all the Hungarian highways. The dataset contained data from October the 1st to October the 14th, 2016 regarding congestions with a length of at least 3 kilometers. Table 4 shows the validation data and the results of the proposed framework.

Setting the number of classes to 10. We define a correct detection when there is at least 50% overlap (in terms of time) over the validation data with traffic classification level equal to 9 or 10. With this setup eleven out of fourteen (79%) congestions have been successfully detected. The time series have been quantized with time windows of 15 minutes, thus during the testing phase the classification takes place each fifteen minutes.

It is interesting to note that the majority of the anomalies are detected earlier than the data provided by utinform. This can be a proof that the proposed solution is able to spot congestions when these are shorter than three kilometers. The outcome is similar for what concerns the end of the congestions, in this case the proposed solution tends to detect the end of an anomaly later than the validation data set. Three congestions out of fourteen have not been detected, however this can be due to at least two reasons: the telecom operator (because of industrial secrecy concerns) did not provide to us any detail regarding the range of antennas or their direction, thus, there might be a chance of inaccuracies along the phase of matching the geographical maps with the towers' positions in order to define the competence of each tower w.r.t. the highways segments. Another reason could be due to data inconsistencies. In fact, as mentioned in Section 2, one third of the data have been dropped.

Figure 4 represents the classification value over time slices of fifteen minutes for the highway segment suffering for the congestion described in Table 4 on row three. The red (in black and white print: the dark) vertical line represents the time slot when the anomaly is detected on the validation set. Here, a congestion is defined when the classification values are equal or greater than 9. On the other end, Figure 5 represents the number of handsets (an estimation of the number of travelers) involved in the congestion.

The application framework have been completely developed in `Python 3.6` and it have been demonstrated to be very efficient. Although the system is designed to work on-line receiving a stream of data in real-time, in order to measure the performances of the model, we decided to exclude the streaming and the networking module, finally we tested the model in off-line mode. One log per time is read and suddenly processed. With this approach we

[3] `http://utinform.hu/`

Cong. Start Time	Cong. End Time	Det. Start Time	Det. End Time
11:00	11:40	10:30	12:15
06:35	07:45	07:15	07:45
09:30	22:10	09:00	00:00
17:55	09:05	none	none
06:52	07:30	07:00	08:15
14:50	16:50	16:45	17:30
08:40	18:10	08:15	18:45
14:00	17:10	14:45	19:25
10:00	11:20	08:45	13:45
17:23	20:15	16:15	20:00
07:20	08:50	07:30	10:30
02:55	04:40	none	none
15:00	16:00	15:15	16:15
06:45	07:25	none	none

Table 4: Experimental Results indicating congestion (Cong.) start and end times as well as detection (Det.) start and end times.

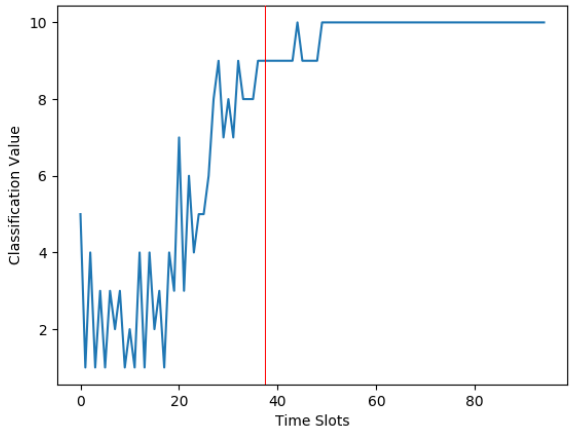


Figure 4: Classification values over time slices and the time of a congestion (red/dark line), an illustrative example.

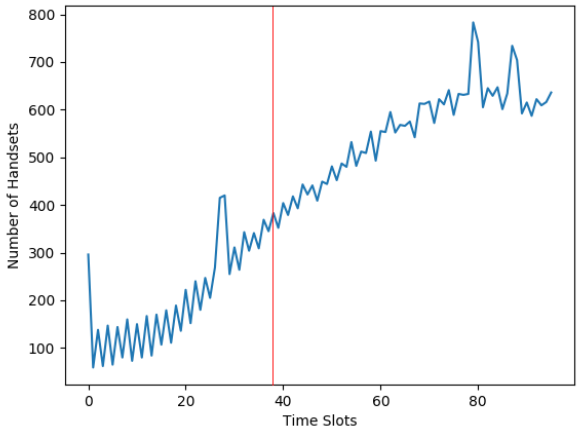


Figure 5: Number of handsets involved in a congestion and the time of the congestion (red/dark line), an illustrative example.

maintained the architectural design of the framework and no-delay straming process has emulated while at the same time avoiding networking related delays. Running the application on an *Intel i7 6700K* with 16GB of DDR4 RAM, on average, manages to process the logs for an entire day within less than 10 minutes with a peak of RAM consumption of 32MB. Furthermore, the application framework have been deployed on a `Raspberry Pi 3` where the running time needed to process an entire set of logs representing one day is in around 1 hour, in average.

5.1 Visualization

First the communication interface need to be mentioned between the back end and the front end part. The front end is a web based application, it uses `HTML` and `JavaScript`, which communications with the Python back end through Web Sockets with `.json` files. At the initialization stage, first the front end asks the back end for mapping of tower IDs and highway sections, as it was mentioned before at the geographic data section. After the initial step, the back end is sending messages which the front end processes and shows on the map. These messages are json files, containing a list of objects with three attributes: ID, value, anomaly. The ID stands for the tower IDs and the value is an int from 0 to 9 showing the traffic load on that segment, and the anomaly flag is giving information that this value is the expected for the given time window or it is deviating from the usual traffic load on that area. In order to keep the low cost functionality of the system, the visualization had

to be carefully built as well. An open-source JavaScript library, `LeafletJS`[4] was used to place layers on the particular highway segments. This library has relatively low cost of handling layers and as it is expected from an application where the traffic load is constantly changing this was a critical feature. Each layer is a colored visualization of the value given to the particular highway section, an example can be seen on the Figure 6, where the spectrum is from blue to red, representing the low to high traffic load.

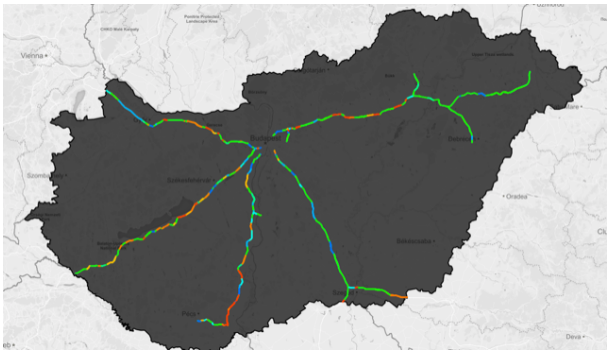


Figure 6: The traffic load visualized on the highway segments, an illustrative example.

6 Conclusions

A lightweight traffic monitoring and traffic jam detection framework has been presented in this paper based on HashMap data structures and methods for breakpoint detection well-known in time series classification. The used concepts require cheap computation and, basically, minimal tuning phase opposite to the case of tuning the hyperparameters of machine learning algorithms. The few parameters of the framework such that the time window or time frames as well as the number of breakpoints can be set according to an available domain knowledge or user expertise. However, to avoid false positives, it is recommended to tune the presented framework before implementing it into a production environment.

The SAX representation had already been used as a tool for time series classification. However, the proposed discretization procedure is unique in that it uses an intermediate representation between the raw time series and the symbolic strings. Furthermore the aim is not to classify full time series as in [5] but rather to classify new incoming single values.

The presented framework was tested using real data. Due to the sensitive nature of the project the presented framework was developed within, the authors cannot disclose the source code nor the data used in experiments, in this time. Experiments show that the proposed framework is promising and worth further development and adaptation to other use-case scenarios.

[4] `http://leafletjs.com`

Acknowledgements

Authors would like to thank Magyar Telekom Nyrt. and utinform.hu. The research has been supported by the European Union, co-financed by the European Social Fund EFOP-3.6.3-VEKOP-16-2017-00001 and the project "Open City services" funded by Magyar Telekom Nyrt.

References

[1] R. Finkel, J.L. Bentley (1974). Quad Trees: A Data Structure for Retrieval on Composite Keys. Acta Informatica 4 (1), 1–9.

[2] Lin, J., Keogh, E., Lonardi, S. and Chiu, B., 2003. A symbolic representation of time series, with implications for streaming algorithms. In Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery (pp. 2-11). ACM.

[3] Lin, J., Keogh, E., Wei, L. and Lonardi, S., 2007. Experiencing SAX: A Novel Symbolic Representation of Time Series. Data Mining and knowledge discovery, 15(2), pp.107-144.

[4] Zhang, Y. and Glass, J., 2011. A piecewise aggregate approximation lower-bound estimate for posteriorgram-based dynamic time warping. 12th Conference of the International Speech Communication Association, pp.1909-1912.

[5] Senin, P. and Malinchik, S., 2013, December. Sax-vsm: Interpretable time series classification using sax and vector space model. In Data Mining (ICDM), 2013 IEEE 13th International Conference on (pp. 1175-1180). IEEE.

[6] Milani, A., Gentili, E. and Poggioni, V., 2009. Cellular Flow in Mobility Networks. IEEE Intelligent Informatics Bulletin, 10(1), pp.17-23.

[7] Hakkarainen, A., Werner, J., Costa, M., Leppanen, K. and Valkama, M., 2015, September. High-efficiency device localization in 5G ultra-dense networks: Prospects and enabling technologies. In Vehicular Technology Conference (VTC Fall), 2015 IEEE 82nd (pp. 1-5). IEEE.

[8] R. Khokale, A. Ghate (2017). Data Mining for Traffic Prediction and Analysis using Big Data. International Journal of Engineering Trends and Technology 48 (3).

[9] Gundlegard, D. and Karlsson, J.M., Road Traffic Estimation using Cellular Network Signaling in Intelligent Transportation Systems, 2009. In: Wireless technologies in Intelligent Transportation Systems. Editors: Ming-Tuo Zhou, Yan Zhang and L. T. Yan. ISBN: 978-1-60741-588-6 2009 pp. 361-392. Nova Science Publishers.

[10] White, J. and Wells, I., 2002. Extracting origin destination information from mobile phone data. In 11th International Conference on Road Transport Information and Control, London, 2002, pp. 30–34

[11] Wideberg, J.P., Caceres, N. and Benitez, F.G., 2006. Deriving Traffic Data from a Cellular Network. In Procedings of the 13th ITS World Congress, London, 2006.

S. Krajči (ed.): ITAT 2018 Proceedings, pp. 116–123
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Rudolf J. Szadkowski, Petr Čížek, and Jan Faigl

Learning Central Pattern Generator Network with Back-Propagation Algorithm

Rudolf J. Szadkowski[1], Petr Čížek[1], and Jan Faigl[1]

Czech Technical University in Prague, Technicka 2, 16627 Prague, Czech Republic,
szadkrud,petr.cizek,faiglj@fel.cvut.cz

An adaptable central pattern generator (CPG) that directly controls the rhythmic motion of multilegged robot must combine plasticity and sustainable periodicity. This combination requires an algorithm that searches the parametric space of the CPG and yields a non-stationary and non-divergent solution. We model the CPG with the pioneering Matsuoka's neural oscillator which is (mostly) non-divergent and provides constraints ensuring non-stationarity. We embed these constraints into the CPG formulation which we further implemented as a layer of an artificial neural network. This enables the CPG to be learnable by back-propagation algorithm while sustaining the desirable properties. Moreover, the proposed CPG can be integrated into more complex networks and trained under different optimization objectives. In addition to the theoretical properties of the developed system, its flexibility is demonstrated in successful learning of the tripod motion gait with its practical deployment on the real hexapod walking robot.

1 Introduction

The movement of legged robots relies on synchronized control of each its joint. Since these joints are part of the same body, the velocity of each joint is dependent on the position of all robot's joints. The problem of generating such synchronized control signals gets harder with increasing number of legs (or the number of joints per leg). A widely used generator of such signals is a system of interconnected Central Pattern Generators (CPGs). The system based on CPGs can be described as two or more coupled oscillators. CPGs appear in many vertebrates and insects where they are responsible for controlling rhythmic motions, such as swimming, walking or respiration [1, 2]. It also appears in biologically inspired robotics, where CPGs are used for locomotion control of legged robots [3].

A CPG network can be modeled as a non-linear dynamic system with coupled variables. Such a non-linear dynamic system is parameterized in the way that it contains a stable limit cycle, but finding such a parametrization is difficult because an analytical description of the high-dimensional non-linear dynamic system is hard or impossible. Moreover, even a small change in the parameters can result in a sudden change of the system's qualitative properties that can range from chaotic to stationary and somewhere between is the desired periodic behavior.

Parameters of the CPG networks can be found experimentally (i.e., tuned manually or automatically by evolutionary algorithms [4]) or they can be heuristically designed. Such design-dependent methods make CPG networks difficult to scale on other robotic bodies or adapt to the locomotion control in different environments. The scaling problem can be partially bypassed by precomputing a trajectory for each foot tip and employing inverse kinematics to determine the control signals for the particular leg's joints [5, 6]. However, the inverse kinematic depends on the robot's body, and identification of the parameters that have to be manually fine-tuned to ensure a proper behavior.

The motivation for the presented approach is to develop a fully automatic CPG learning and this paper explores the possibility of learning a CPG network modeled by Matsuoka's neural oscillators [7] with back-propagation algorithm (BP). To boost the BP algorithm that learns the desired locomotion control for our multi-legged walking robot, we propose two methods pruning the parameter space of the CPG network.

The particular contributions presented in the paper are considered as follows.

- A normalization layer that prunes the parameter space from parametrization with stable stationary solutions.
- An inductive learning method that exploits the structure of robot's body and further reduces the searched parametric space.
- Experimental evaluation of the proposed learning using real hexapod walking robot for which the proposed CPG network learned by the designed algorithm exhibits successful locomotion control following tripod gait, where the developed CPG network directly produces the control signal for each of 18 actuators of the robot.

2 Related Work

Different biomimetic approaches including CPGs [1], Recurrent Neural Networks [8] or Self-Adjusting Ring Modules [9] to produce rhythmic patterns have been studied and deployed for locomotion control of robots [3] in recent years. These approaches differ mainly in the complexity of

the underlying model and have different levels of abstraction ranging from biomechanical models [10] simulating membrane potentials and ion flows inside neurons, down to a model of two coupled neurons in a mutual inhibition [11]. Amongst them, the CPGs based on Matsuoka's neural oscillator [7] are being used as the prevalent model. Further details on the Matsuoka's model are in Section 3 as we built on its properties [7, 12, 13] in our work.

Deployment of the CPG oscillators on legged robots is also particularly difficult because of different kinematics and dynamics of each robot. A different amount of post-processing is used to translate the CPG outputs to joint coordinates. Namely, approaches using inverse kinematics [5, 6] suffer from necessary hand fine-tuning of both the parameters of CPG as-well-as kinematics. Besides, existing approaches are using the separate neural network as motor control unit [11] or use CPG outputs directly as joint angles [14]. Furthermore, CPGs can seamlessly switch between different output patterns, thus different gaits [15] which further supports the direct joint control. In our work, we use a dedicated output layer to shape the outputs of CPGs as we assume simple transformations of the output signal are easier to learn by changing parameters of the output layer while the gait change is in charge of the CPG.

Parametrization of the oscillator can be found experimentally, e.g., using evolutionary algorithms with fitness function minimizing energy consumption [11], maximizing the velocity [4], or using parameter optimization [16]. Besides, a modified back-propagation algorithm has been used on an adaptive neural oscillator in [17] to imitate an external periodic signal by its output signal, but it fails to sustain oscillations for complex waveforms. Further works on the parameter constraining of CPGs to maintain stable oscillations have been published [7,12,13,16]; however, to the best of our knowledge we are the first to teach a network of CPGs to perform a locomotion gait of a hexapod walking robot using back-propagation. Furthermore, we propose two methods to prune the space of possible CPG parameters.

3 Central Pattern Generator Network

The CPG network used in this paper is based on the Matsuoka's neural oscillator [7]. Matsuoka's neural oscillator is a pair of symmetrically connected adaptive neurons, extensor, and flexor, that imitate the behavior of biological neurons where after peaking, the neuron starts to repolarize until its activation drops to resting potential. Features of Matsuoka's neurons were extensively studied; hence, necessary conditions under which the neural network enters the stable stationary state [7], effects of time-variant tonic input [12], and approximation of oscillator's fundamental frequency and amplitude [13] are well documented in the literature. The description of the particular CPG model used in this work is as follows.

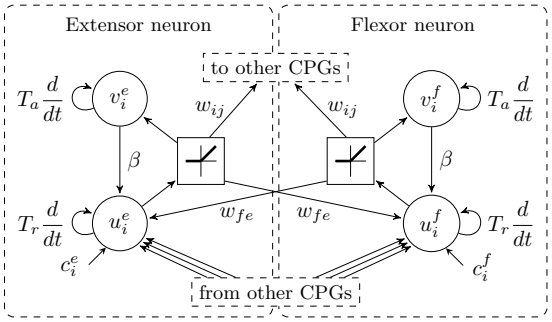


Figure 1: CPG unit connected to the CPG network.

3.1 CPG Model

The dynamics of the CPG network with N units can be described by a set of equations

$$T_r \dot{u}_i^e = -u_i^e - w_{fe} g(u_i^f) - \beta v_i^e - \sum_{j=1}^{N} w_{ij} g(u_j^e) + c_i^e, \quad (1)$$

$$T_a \dot{v}_i^e = g(u_i^e) - v_i^e, \quad (2)$$

$$T_r \dot{u}_i^f = -u_i^f - w_{fe} g(u_i^e) - \beta v_i^f - \sum_{j=1}^{N} w_{ij} g(u_j^f) + c_i^f, \quad (3)$$

$$T_a \dot{v}_i^f = g(u_i^f) - v_i^f, \quad (4)$$

where the subscript $i \in N$ denotes the particular CPG and the superscript $\mu \in \{e, f\}$ distinguishes the extensor and flexor neurons, respectively. Each tuple of the variables u_i^e, v_i^e describes the dynamics of the extensor neuron. The variable u_i^e represents activation of the neuron and v_i^e represents its self-inhibitory input, which makes this neuron adaptive. Similarly u_i^f, v_i^f describe the dynamics of the flexor neuron. The function g is a rectifier

$$g(x) = \max(0, x) \quad (5)$$

that is an activation function that adds non-linearity to the system. Each neuron (i, μ) inhibits itself through the variable v_i^μ scaled by the parameter $\beta > 0$. The extensor-flexor pair (i.e., the CPG unit) mutually inhibits itself through the symmetric connection with the weight $w_{fe} > 0$. Finally, the CPG units are inter-connected with the symmetric inhibiting connections $w_{ij} \in W$ for $w_{ij} \geq 0$ and $w_{ii} = 0$, where W is a symmetric matrix. The only source of excitation for this CPG network is the tonic input c_i^e, c_i^f (≥ 0) which is given externally. In general, the tonic input may be time-dependent and can be used to regulate the output of the CPG network [12]. T_r and T_a (both > 0) are reaction times for their respective variables. The structure of the CPG unit is visualized in Fig. 1.

All the equations (1), (2), (3), and (4) are differentiable except the cases when $u_i^\mu = 0$, since the rectifier is used as the activation function. However, we assume this will not cause any problems because the rectifier is used inside the Rectified Linear Units (ReLU), which are widely used in deep neural networks.

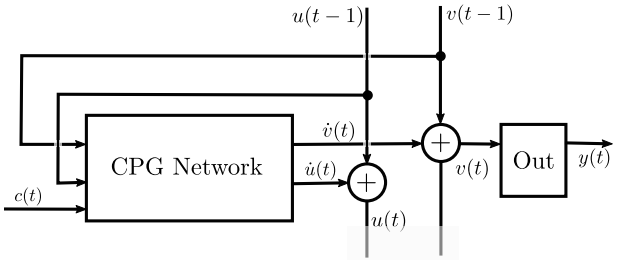
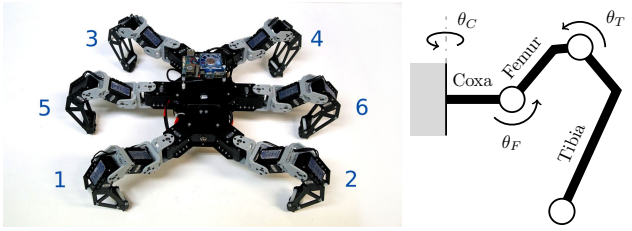


Figure 2: The CPG network connected to the output layer `Out`. Notice the output y is not fed back to the network. Also notice that the self-inhibitory input u is not connected to `Out`.



(a) (b)

Figure 3: (a) Hexapod robot with the numbered legs. (b) Schema of the leg. Each leg consists of three parts – Coxa, Femur, and Tibia.

Note that except tonic inputs c_i^e, c_i^f, there are used only inhibiting connections, because such a system is less prone to become chaotic or divergent [13].

3.2 Output layer

In this work, we consider the self-inhibitory inputs v^e, v^f as hidden variables, we do not work with them outside of the CPG network. The output layer combines the activation variables u^e, u^f with the affine transformation

$$\mathbf{y} = W_{out}\mathbf{u} + \mathbf{b}_{out}, \tag{6}$$

where $\mathbf{u} = (\mathbf{u}^e, \mathbf{u}^f)$ and $W_{out} \in \mathbb{R}^{N\times 2N}, \mathbf{b}_{out} \in \mathbb{R}^{N\times 1}$ are the learnable parameters. The connection of the CPG network and the output layer is illustrated in Fig. 2.

The main advantage of having W_{out} and $\mathbf{b}_{out}$ as learnable parameters are that the BP algorithm can scale and translate the limit cycle formed by the CPG network. Here, we assume that these transformations are easier to learn by changing the parameters of the output layer than by changing parameters of the CPG network. It is because a change of any parameter of the CPG network can generally cause a non-linear change in the amplitude, frequency, and shift of the generated signals [6]. Another advantage of the proposed output layer is that it can develop complex signals as it can combine outputs from different CPGs.

4 Proposed Locomotion Control Learning

In this section, we propose the normalization layer and inductive learning method adapted to learning a CPG network for a hexapod walking robot, see Fig. 3a. Each leg of the robot has three joints called coxa, femur, and tibia (see Fig. 3b) for which an appropriate control signal has to be generated to control the locomotion of the robot. In the total, the robot has 18 controllable joints and depending on the control signals; the robot can move with various motion gaits [18], e.g., tripod, quadruped, wave, and pentapod. During the locomotion, each leg is either in a swing phase to reach a new foothold or in the stance phase in which it supports the body. The motion gait prescribes the order in which the swing and support phases alternate for individual legs; hence, all the legs must work in coordination to simultaneously achieve the desired behavior. The hexapod walking robot is thus used for benchmarking the proposed learning method, where the CPG network has to learn to generate control signals that realize the locomotion control of the robot with the tripod motion gait.

4.1 Normalization layer

The proposed normalization layer is based on early experiments with randomly parametrized CPG networks which in most cases ends up oscillating or converges to a static behavior. The static behavior is caused by the stable fixed points that may appear in the corresponding dynamic system. Therefore, we propose to employ a sufficient condition for the CPG network to be free of stable fixed points.

Condition. *For a CPG network of N units, if all the values of the tonic input c_i^μ, where $i \in N$ and $\mu \in \{e, f\}$, are from the range $[c_{min}, c_{max}]$ and*

$$w_{fe} < \frac{c_{min}}{c_{max}}(1+\beta) - \max_{i\in N}\left(\sum_j^N w_{ij}\right), \tag{7}$$

$$w_{fe} > 1 + T_r/T_a \tag{8}$$

then the CPG network has no stable fixed point.

Proof. First, we state adapted theorem from [7].

Theorem. *Assume that for some i and k ($i \neq k$)*

$$c_i(1+\beta) - \sum_j^{2N} a_{ij}c_j > 0, \tag{9}$$

$$c_k(1+\beta) - \sum_j^{2N} a_{kj}c_j > 0, \tag{10}$$

$$a_{ik} > 1 + T_r/T_a, \tag{11}$$

then the CPG network has no stable fixed point. The term $\{a_{ij}\} = A^{(2N,2N)}$ is a matrix of the form

$$A = \begin{bmatrix} W & w_{fe}I \\ w_{fe}I & W \end{bmatrix} \tag{12}$$

and $\mathbf{c} = (c^e, c^f)$*, where I is the identity matrix of the same dimensions as* W.

Since the CPGs should act as independent units, it is intuitive that each extensor-flexor neuron pair (a CPG) is able to oscillate on its own. Thus, a weaker form of the theorem is used, where the following conditions must hold for each i-th CPG:

$$\frac{c_i^e}{c_i^f}(1+\beta) - \frac{1}{c_i^f}\sum_j^N w_{ij}c_j^e > w_{fe} \tag{13}$$

$$\frac{c_i^f}{c_i^e}(1+\beta) - \frac{1}{c_i^e}\sum_j^N w_{ij}c_j^f > w_{fe} \tag{14}$$

$$w_{fe} > 1 + T_r/T_a. \tag{15}$$

Now, we can focus on the effect of the tonic input c. For any parametrization $W, \beta, T_r, T_a, w_{fe}$ we can find a vector c that would break these conditions. Let's relax the problem by clipping the values of $\mathbf{c}$ into the range $[c_{min}, c_{max}]$ where $c_{min} > 0$. Then, it must become independent on the mutable $\mathbf{c}$ vector to simplify the system of conditions. This can be done by substituting $\mathbf{c}$ with such $\mathbf{c}_i^-$ that minimizes the left side expression of (13) or (14) for the i-th CPG. W.l.o.g. we consider finding $\mathbf{c}_i^-$ just for (13) as

$$\mathbf{c}_i^- = \underset{\mathbf{c}\in[c_{min},c_{max}]^{2N}}{\operatorname{argmin}} \frac{c_i^e}{c_i^f}(1+\beta) - \frac{1}{c_i^f}\sum_j^N w_{ij}c_j^e. \tag{16}$$

Since all the parameters are positive and $w_{ii} = 0$, the min argument in (16) decreases monotonically with decreasing c_i^e and increasing c_j^f values. Thus, we can substitute these variables with their respective extremes

$$\mathbf{c}_i^- = \begin{cases} c_j^e \in \mathbb{R}^+, j \neq i \\ c_i^e = c_{min} \\ c_j^f = c_{max}, j \neq i \\ c_i^f = c_i' \end{cases} \tag{17}$$

that leaves just c_i' as the variable to minimize

$$F(c) = \frac{c_{min}}{c}(1+\beta) - \frac{c_{max}}{c}\sum_j^N w_{ij}, \tag{18}$$

$$c_i' = \underset{c_i^f \in [c_{min},c_{max}]}{\operatorname{argmin}} F(c_i^f). \tag{19}$$

Notice that now, we are searching a scalar value c_i' that minimizes the given expression.

The equation $\frac{dF(c)}{dc} = 0$ has a solution only if F has such parameters β, W, c_{min}, and c_{max} that make the function F constant. Since it is unlikely that such a parametrization will emerge during the learning, we consider F does not have any local extremes in the range $[c_{min}, c_{max}]$. Therefore, the minimization (19) can be simplified to

$$c_i' = \operatorname{argmin}\{F(c_{min}), F(c_{max})\}. \tag{20}$$

The condition (13) implies $F > 0$, because w_{fe} must be greater than zero and the following condition must hold too

$$1 + \beta > \frac{c_{max}}{c_{min}}\sum_j^N w_{ij}. \tag{21}$$

Now, we define variable $\varepsilon > 0$ that

$$1 + \beta = \frac{c_{max}}{c_{min}}\sum_j^N w_{ij} + \varepsilon \tag{22}$$

and substitute the right side of (22) into $F(c_{min})$ and $F(c_{max})$

$$F(c_{max}) = \frac{c_{min}}{c_{max}}\varepsilon, \tag{23}$$

$$F(c_{min}) = \varepsilon. \tag{24}$$

Since $\frac{c_{min}}{c_{max}} \in (0, 1]$ and $\varepsilon > 0$, the expression $F(c_{max})$ always minimizes (20). Therefore

$$c_i' = c_{max}. \tag{25}$$

After substituting c_i' into (17) and then $\mathbf{c}_i^-$ into (13) we get

$$\frac{c_{min}}{c_{max}}(1+\beta) - \sum_j^N w_{ij} > w_{fe}. \tag{26}$$

Finally, to make this condition independent on the i-th CPG, we can choose such an inequality (26) that has the largest value of the $\sum_j^N w_{ij}$ expression

$$w_{fe} < \frac{c_{min}}{c_{max}}(1+\beta) - \max_{i\in N}\left(\sum_j^N w_{ij}\right). \tag{27}$$

Combining (15) and (27) we get the desired (8) and (7). ■

We integrate the conditions (7) and (8) into the BP framework by redefining the variables w_{fe} and β as functions

$$w_{fe}(\hat{w}_{fe}, T_r, T_a) = 1 + T_r/T_a + \exp(\hat{w}_{fe}), \tag{28}$$

$$\beta(\hat{\beta}, w_{fe}, w^*) = (w_{fe} + w^*)\frac{c_{max}}{c_{min}} + \exp(\hat{\beta}) - 1, \tag{29}$$

where $\hat{w}_{fe}, \hat{\beta} \in \mathbb{R}$ are new independent parameters and w^* is defined as

$$w^* = \max_{i\in N}\left(\sum_j^N w_{ij}\right). \tag{30}$$

Then, the max operator is approximated by the differentiable smoothmax defined as

$$\operatorname{softmax}(x) = \frac{\exp(x)}{\sum\exp(x)}, \tag{31}$$

$$\operatorname{smoothmax}(x) = \operatorname{softmax}(x)x. \tag{32}$$

Since all the parameters must be positive, other parameters are defined as exponent of the underlying parameter as

$$\begin{aligned} T_a &= \exp(\hat{T}_a), \\ T_r &= \exp(\hat{T}_r), \\ w_{ij} &= \exp(\hat{w}_{ij}), i \neq j, \end{aligned} \tag{33}$$

where $\hat{T}_a, \hat{T}_r, \hat{w}_{ij} \in \mathbb{R}$. The weights $w_{ij}, i \neq j$ cannot reach zero during learning, but they can approach it.

The BP algorithm learns the proposed new parameters $\hat{T}_a, \hat{T}_r, \hat{w}_{ij}, \hat{w}_{fe}$, and $\hat{\beta}$ that are later normalized by (28), (29), and (33).

4.2 Proposed Architecture and Inductive Learning

We propose to divide the CPG network into smaller sub-networks to reduce the search parameter space. These sub-networks are independently learned and then merged into larger sub-networks until a single final network remains. The proposed learning of the CPG network is performed in three phases. First, we learn a single CPG to generate a signal for one joint which gives us the shared parameters $(w_{fe}, T_a, T_r, \beta)$. Then, six triplets of CPGs are learned to generate a control signal for the particular leg. Therefore, for each leg $k \in [1, \ldots, 6]$, we get parameters W^k and $W^k_{out}, \mathbf{b}^k_{out}$. In the final phase, we connect all six CPG sub-networks into one. We choose to connect CPG sub-networks only by coxa-CPGs as it is assumed this is enough for each CPG sub-network to synchronize. Therefore, for the subspace $\mathbf{u}^e = (u^e_{coxa,1}, \ldots, u^e_{coxa,6}, u^e_{femur,1}, \ldots, u^e_{tibia,1})$ (and similarly for $\mathbf{u}^f$), $W \in \mathbb{R}^{18\times 18}$ is organized as follows

$$W = \begin{bmatrix} W_{coxa,coxa} & W_{coxa,femur} & W_{coxa,tibia} \\ W_{femur,coxa} & 0 & W_{femur,tibia} \\ W_{tibia,coxa} & W_{tibia,femur} & 0 \end{bmatrix},$$

where $W_{ij}, i \neq j$ is the matrix of the connections between the i-th and j-th joints that can be expressed as

$$W_{ij} = \begin{bmatrix} w^1_{ij} & 0 & 0 \\ 0 & \cdots & 0 \\ 0 & 0 & w^6_{ij} \end{bmatrix},$$

where the weights $\{w^k_{ij}\} = W^k$ are taken from the matrices parametrizing the previously learned CPG sub-networks.

For the rearranged vector $\mathbf{u} = (u^e_1, u^f_1, \ldots, u^e_6, u^f_6)$, the term $W_{out} \in \mathbb{R}^{18\times 36}$ is composed of the matrices W^k_{out} of the previously learned CPG network that controls the k-th leg

$$W_{out} = \begin{bmatrix} W^1_{out} & 0 & 0 \\ 0 & \cdots & 0 \\ 0 & 0 & W^6_{out} \end{bmatrix}.$$

All the zeroes in the W and W_{out} matrices are unlearnable constants imposing a structure onto the CPG network.

4.3 Objective Function

The utilized loss function of the CPG network is defined as a positive distance of the output vector from the desired one

$$\mathscr{L}(\mathbf{y}(t), \mathbf{d}(t)) = \|\mathbf{y}(t) - \mathbf{d}(t)\|, \tag{34}$$

where $\mathbf{d}(t) \in [0,1]^{18}$ is the target signal for each of 18 robot's actuators at the time t.

During early evaluation of the proposed learning, we observed that in many cases, the output signal has undesired lower frequency harmonics. This caused the output signal to fit the target signal only for a couple of the first periods. We propose to address this issue by an additional term to the objective function (34)

$$+ \|r - \omega\|, \tag{35}$$

where $r \in \mathbb{R}^+$ is a new hyperparameter and ω is an approximation of the fundamental frequency of the CPG oscillations that can be expressed as [13]

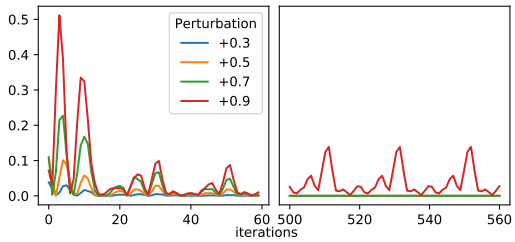
$$\omega = \frac{1}{T_a}\sqrt{\frac{(T_r + T_a)\beta - T_r w_{fe}}{T_r w_{fe}}}. \tag{36}$$

The hyperparameter r should be equal to the fundamental frequency of the desired signal. However, since (36) is just an approximation; it might lead to undesired local minima. Therefore, we propose to switch off the regularization once the term (35) is lesser than a predefined threshold.

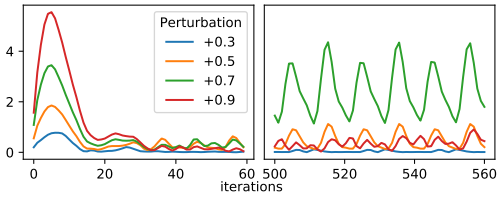
5 Experimental evaluation

The proposed learning method has been experimentally verified using `rmsprop` [19] algorithm, which is commonly used to learn recurrent neural networks. Since the following experiments are meant to benchmark and map problems of the CPG network learning, we use a constant tonic input $\mathbf{c} = \mathbf{1}$. Therefore, $c_{min} = c_{max} = 1$. The initial state $(\mathbf{u}^e_{init}, \mathbf{v}^e_{init}, \mathbf{u}^f_{init}, \mathbf{v}^f_{init})$ is set to $\mathbf{u}^e_{init} = 0.1$, $\mathbf{u}^f_{init} = -0.1$, and $\mathbf{v}^f_{init} = \mathbf{v}^f_{init} = 0$. The target signal is formed of eighteen sequences of joint angles that were recorded for a course of five tripod gait cycles. The hexapod robot was driven by a default regular gait based on [20], which is suitable for traversing flat terrains, and it uses the inverse kinematics for following the prescribed triangular leg foot-tip trajectory. This 4.7 seconds long record of all joint signals is sampled to 2350 equidistant data points, and each signal is further normalized in the range $[0,1]$, smoothed using Gaussian convolution to filter out signal peaks, and finally downsampled by the factor of 3.

Preliminary experiments have shown that the process of learning profoundly depends on initial parameters and in some runs, the BP algorithm seems to stuck in local minima from which the learning becomes very slow. This observation is consistent with [17]. The performance of the



(a)



(b)

Figure 4: Squared signal errors of the first leg (a) and body (b) caused by perturbations. The perturbations have been introduced only at the start by adding a constant value to all the trajectory components. For the leg CPG network (a) after 500 iterations, the errors vanished except for +0.9 perturbation. The body CPG network (b) does not recover even after 500 iterations except for +0.3 perturbation.

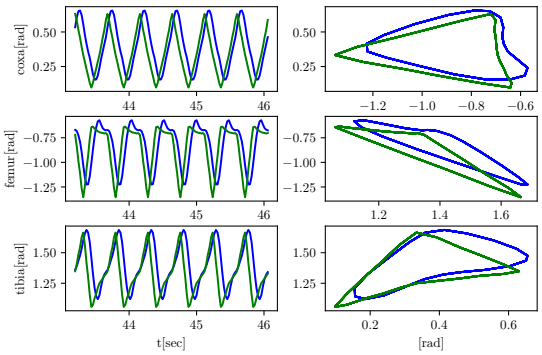


Figure 5: Signals controlling the joints of the first leg. Green ones are desired, and blue ones are generated by the CPG network. The time evolution is on the left, while projected phase-space trajectories are on the right. Each row corresponds to one joint, i.e., coxa, femur, and tibia. In the phase-space column, the variable pairs from the top are coxa-femur, femur-tibia, and tibia-coxa.

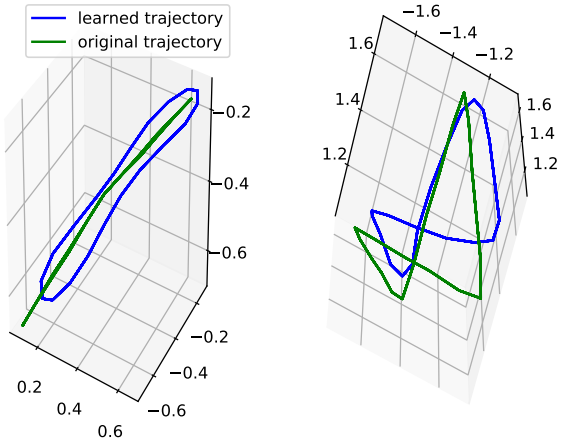


Figure 6: Synchronization of multiple legs. On the left, the coxa-control trajectory for legs 1, 2, and 3 depicted in Fig. 3a. The original trajectory moves almost diagonally in the pictured cube and is "wrapped" by the learned trajectory. On the right, the tibia-control trajectory for the legs 1, 2, and 3.

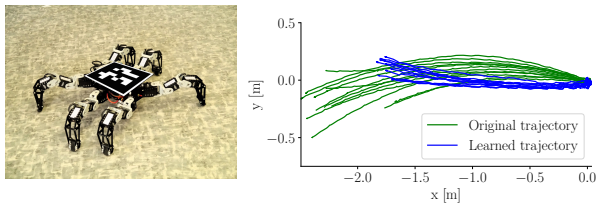
BP algorithm has been improved by adding the regularization term (35). After that, the learning is performed in the three following consecutive steps.

First, each single CPG unit is learned to generate the sinusoid $\sin(t/2)$ that has the same frequency as the fundamental frequency of the desired control signal, which is deterministically set to 3 Hz. The CPG is learned in 2000 epochs, each back-propagating a batch of size 50 data-points. Note that the number of the needed epochs depends on the initial random parametrization.

Next, the parameters of the sinusoid generator is retrained to generate the desired joint control signals. The generator of each joint control is learned with 2000 epochs. We experimented with the stability of the learned limit cycle of the first leg by perturbing it, see Fig. 4a. Finally, the joints CPGs are connected as described in Sec. 4 with non-diagonal values of $W_{coxa,coxa}$ initialized to 0.5, and learned with 4000 epochs. We experimented with the stability of this final CPG network and results are depicted in Fig. 4b.

A comparison of the desired control signal of the first leg and the learned signal is depicted in Fig. 5. The learned signal has a similar shape and the same frequency as the original signal. Binding between different triplets of the legs, the most difficult part is shown in Fig. 6. We can see that the learned trajectory has a similar structure to the desired limit cycles. The trajectory also stays within its limit cycle; the trajectory was generated by six gait-cycles, therefore, traveled the limit cycle multiple times.

We deployed the resultant CPG locomotion controller on the real hexapod (see Fig. 3a) and compared with the original controller [20] in 10 trials. The robot was requested to crawl on flat surface for 10 s and then stop. The velocity of the robot was estimated using an external visual localization system based on tracking of visual marker [21] running with 25 Hz. Moreover, the robot's



(a) (b)

Figure 7: (a) Used experimental setup of the robot with the tracking marker. (b) Visualization of the performed trajectories using the proposed CPG locomotion control and the reference controller.

stability was measured as smoothness of the locomotion using an XSens MTi-30 inertial measurement unit (IMU) attached to the robot trunk. The variances in vertical acceleration (Acc_z) and the orientation (pitch and roll angles) of the robot's body are the selected indicators of the locomotion stability.

The recorded robot trajectories visualized in Fig. 7 show that there is a transition effect for our CPG locomotion controller at the beginning of the trajectory where the CPG network starts to oscillate which makes the robot initial acceleration lower; however, the overall locomotion is smoother, as the velocity deviation is smaller.

The quantitative results are listed in Table 1 as average values of the indicators. The results indicate that the performance of the CPG locomotion controller is similar to the implementation [20] based on inverse kinematics (IKT).

Table 1: Experimental results

	Unit	**IKT** [20]	**CPG** (Ours)
Velocity	$[\mathrm{m{\cdot}s^{-1}}]$	0.18±0.03	0.15±0.01
Acc_z var.	$[\mathrm{m{\cdot}s^{-2}}]$	18.31	23.03
Pitch var.	$\times 10^{-3}$[rad]	0.14	0.23
Roll var.	$\times 10^{-3}$[rad]	0.25	0.28

5.1 Lessons Learned and Discussion

During the experimental evaluation of the proposed learning of the CPG network, a couple of good practices how to learn the sinusoid generator came up as follows.

1. It is better to learn the network in batches containing at most two periods.
2. If the CPG network is restarted to the initial state, it is good to ignore the transient states.
3. Since it is not important at which place the system enters the limit cycle, it is suitable to phase-shift the target signal; so, to minimize the distance from the output signal.

Combination of sub-networks into one network has two difficulties. The parameters $(w_{fe}, T_a, T_r, \beta)$ must be the same for the whole CPG network, but the sub-networks are trained independently; so, they can end up with different parameters. In our case, the parameters are similar because all the CPG sub-networks are based on one CPG sub-network. Thus, the BP algorithm is able to adjust them during the learning of the complete network. Another difficulty is the choice of the initial $W_{coxa,coxa}$ weights. The higher the weights are, the stronger is the coupling between the legs. However, if the weight values are too high, the constraint (7) would be violated. Therefore, we used (7) to choose the initial $W_{coxa,coxa}$ weights.

Even though that the robustness is not the objective of the learning algorithm, it is a property of single Matsuoka's oscillator [22]. This property translated well into our 3-unit CPG network (see Fig. 4a) where the network can recover from perturbations. In the real world, robustness helps quickly react to simple temporal events, e.g., servo errors, or feedback from the environment.

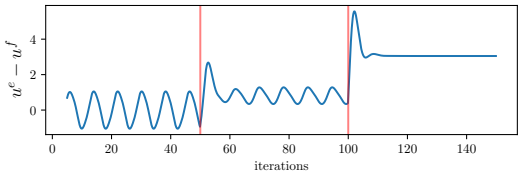
In this work, we chose a simple model with $c_{min} = c_{max} = 1$, i.e., we have a constant tonic input. The time-variant tonic input; however, introduces dynamic changes as we can see in Fig. 8. In the future work, we would like to use the tonic input to control the output of the CPG network dynamically.

6 Conclusion

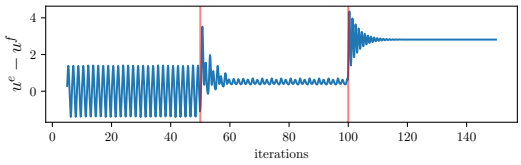
In this paper, we propose a new methodology for learning a CPG network modeled by symmetrically connected neural oscillators. The method is based on a combination of the back-propagation learning algorithm, normalization layer, and regularization term, where the normalization layer prunes the parameters spaces of the CPG network from the undesired non-periodic results, and thus help to speed up the learning process. The advantage of the proposed solution over the previous work on the CPG-based locomotion control is in the scalability of the method that enables to create such a CPG network that can directly control each actuator without the need to employ the inverse kinematics. The proposed method has been successfully deployed in the locomotion control of the real hexapod walking robot.

The main properties of the proposed methodology arise from the idea that the proposed CPG network for the hexapod locomotion control is based on the architecture of the CPG connections that imitates the structure of the robot. The CPG is inductively learned by learning its parts and merging them. Therefore, the proposed method is promising to be easily extendable to other multi-legged robot bodies. Furthermore, since the proposed CPG network is learnable by the back-propagation algorithm, it can be integrated into more complex neural networks supporting back-propagation, which is a subject of our future work.

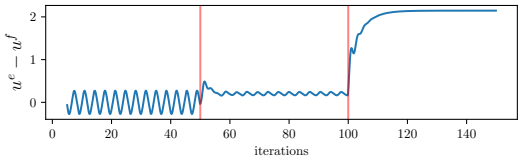
Acknowledgments – This work was supported by the



(a)



(b)



(c)

Figure 8: Output of three randomly generated CPG units with $c_{min} = 2; c_{max} = 4$. After each 50 iterations of total 150 iterations, the tonic input is set to $c^e = c^f = 2$; $c^e = 2, c^f = 4$; $c^e = 2, c^f = 8$, respectively. Note that the last setup violates the c_{max} constraint.

Czech Science Foundation (GAČR) under research project No. 18-18858S. The support of the Grant Agency of the CTU in Prague under grant No. SGS16/235/OHK3/3T/13 to Rudolf Szadkowski is also gratefully acknowledged.

References

[1] E. Marder and D. Bucher, "Central pattern generators and the control of rhythmic movements," *Current Biology*, vol. 11, no. 23, pp. R986–R996, 2001.

[2] E. Marder, D. Bucher, D. J. Schulz, and A. L. Taylor, "Invertebrate central pattern generation moves along," *Current Biology*, vol. 15, no. 17, pp. 685–699, 2005.

[3] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: A review," *Neural Networks*, vol. 21, no. 4, pp. 642–653, 2008.

[4] R. D. Beer, H. J. Chiel, and J. C. Gallagher, "Evolution and analysis of model CPGs for walking: II. General principles and individual variability," *Journal of Computational Neuroscience*, vol. 7, no. 2, pp. 119–147, 1999.

[5] H. Yu, H. Gao, L. Ding, M. Li, Z. Deng, and G. Liu, "Gait Generation With Smooth Transition Using CPG-Based Locomotion Control for Hexapod Walking Robot," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 9, pp. 5488–5500, 2016.

[6] G. Zhong, L. Chen, Z. Jiao, J. Li, and H. Deng, "Locomotion control and gait planning of a novel hexapod robot using biomimetic neurons," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 2, pp. 624–636, 2018.

[7] K. Matsuoka, "Sustained oscillations generated by mutually inhibiting neurons with adaptation," *Biological Cybernetics*, vol. 52, no. 6, pp. 367–376, 1985.

[8] B. A. Pearlmutter, "Gradient calculations for dynamic recurrent neural networks: A survey," *IEEE Transactions on Neural networks*, vol. 6, no. 5, pp. 1212–1228, 1995.

[9] M. Hild and F. Pasemann, "Self-Adjusting Ring Modules (SARMs) for Flexible Gait Pattern Generation." in *IEEE International Joint Conference on Artificial Intelligence (IJCAI)*, 2007, pp. 848–852.

[10] J. Hellgren, S. Grillner, and A. Lansner, "Computer simulation of the segmental neural network generating locomotion in lamprey by using populations of network interneurons," *Biological Cybernetics*, vol. 68, no. 1, pp. 1–13, Nov 1992.

[11] S. Steingrube, M. Timme, F. Wörgötter, and P. Manoonpong, "Self-organized adaptation of a simple neural circuit enables complex robot behaviour," *Nature physics*, vol. 6, no. 3, pp. 224–230, 2010.

[12] K. Matsuoka, "Mechanisms of frequency and pattern control in the neural rhythm generators," *Biological Cybernetics*, vol. 56, no. 5, pp. 345–353, 1987.

[13] ——, "Analysis of a neural oscillator," *Biological Cybernetics*, vol. 104, no. 4, pp. 297–304, 2011.

[14] A. J. Ijspeert, A. Crespi, and J.-M. Cabelguen, "Simulation and robotics studies of salamander locomotion," *Neuroinformatics*, vol. 3, no. 3, pp. 171–195, 2005.

[15] W. Chen, G. Ren, J. Zhang, and J. Wang, "Smooth transition between different gaits of a hexapod robot via a central pattern generators algorithm," *Journal of Intelligent & Robotic Systems*, vol. 67, no. 3, pp. 255–270, Sep 2012.

[16] L. Righetti and A. J. Ijspeert, "Design methodologies for central pattern generators: an application to crawling humanoids," in *Robotics: Science and Systems*, 2006, pp. 191–198.

[17] K. Doya and S. Yoshizawa, "Adaptive neural oscillator using continuous-time back-propagation learning," *Neural Networks*, vol. 2, pp. 375–385, 12 1989.

[18] N. Porcino, "Hexapod gait control by a neural network," in *IEEE International Joint Conference on Neural Networks (IJCNN)*, 1990, pp. 189–194.

[19] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[20] J. Mrva and J. Faigl, "Tactile sensing with servo drives feedback only for blind hexapod walking robot," in *10th International Workshop on Robot Motion and Control (RoMoCo)*, 2015, pp. 240–245.

[21] E. Olson, "AprilTag: A Robust and Flexible Visual Fiducial System," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 3400–3407.

[22] M. M. Williamson, "Robot arm control exploiting natural dynamics," Ph.D. dissertation, Massachusetts Institute of Technology, 1999.

Slovenskočeský NLP workshop (SloNLP 2018)

SloNLP is a workshop focused on Natural Language Processing (NLP) and Computational Linguistics. Its primary aim is to promote cooperation among NLP researchers in Slovakia and Czech Republic.

The topics of the workshop include automatic speech recognition, automatic natural language analysis and generation (morphology, syntax, semantics, etc.), dialogue systems, machine translation, information retrieval, practical applications of NLP technologies, and other topics of computational linguistics.

Workshop Program Committee

Rudolf Rosa, ÚFAL MFF UK, main organizer
Petra Barančíková, ÚFAL MFF UK, main organizer
Vladimír Benko, JÚĽŠ SAV
Ján Genči, KPI TUKE
Aleš Horák, FI MUNI
Miloslav Konopík, KIV ZČU
Pavel Král, KIV ZČU
Markéta Lopatková, ÚFAL MFF UK
David Mareček, ÚFAL MFF UK
Alexandr Rosen, ÚTKL FF UK

S. Krajči (ed.): ITAT 2018 Proceedings, pp. 126–129
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Vladimír Benko

Crowdsourcing for the Slovak Morphological Lexicon

Vladimír Benko
UNESCO Chair in Plurilingual and Multicultural Communication
Comenius University in Bratislava
Šafárikovo nám. 6, SK-81499 Bratilava, Slovakia
and
Ľ. Štúr Institute of Linguistics, Slovak Academy of Sciences
Panská 26, SK-81101 Bratislava, Slovakia

Abstract. *We present an on-going experiment aimed at improving the results of Slovak PoS tagging by means of increasing the size of morphological lexicon that is used for training the respective tagger(s). The frequency list of out-of-vocabulary (OOV) word forms along with the tags and lemmas assigned by the guesser is manually checked, corrected and classified by students in the framework of assignments, so that valid lexical items candidates for inclusion into the morphological lexicon could be identified. We expect to improve the lexicon coverage by the most frequent proper names and foreign words, as well as to create an auxiliary lexicon containing the most frequent typos.*

1 Introduction

"Crowdsourcing" is a relatively recent concept that encompasses many practices. This diversity leads to the blurring of the limits of crowdsourcing that may be identified virtually with any type of Internet-based collaborative activity, such as co-creation or user innovation [1]. In their paper, authors define eight characteristics typical for crowdsourcing as follows:

- *There is a clearly defined crowd (a)*
- *There exists a task with a clear goal (b)*
- *The recompense received by the crowd is clear (c)*
- *The crowdsourcer is clearly identified (d)*
- *The compensation to be received by the crowdsourcer is clearly defined (e)*
- *It is an online assigned process of participative type (f)*
- *It uses an open call of variable extent (g)*
- *It uses the Internet (h)*

From this perspective, language data annotation performed by students in the framework of the end-of-term assignments can well be considered "crowdsourcing", even if only some of the above characteristics apply. It is also worth noting that, according to our experience, students appreciate the feeling that their work may be useful not only as a tool for classification.

2 The Problem

Slovak belongs to languages with more than one system for morphosyntactic annotation available, with two of them being actively used in our work[1] . They have been developed (partially independently) in the framework of two different research projects.

The *Slovak National Corpus* (*SNC*) [2] is using a system based on the new Czech *MorphoDiTa* tagger [3, 4] with a custom language model and a tool for guessing lemmas for unrecognized (out-of-vocabulary – *OOV*) lexical items; while the *Aranea* Project [5, 6] is using a more traditional *TreeTagger* [7, 8] with a custom language model, yet without any functionality to guess lemmas for the *OOV* lexical items. Both systems are using the *SNC* tagset[2] [9] – a fine-grained positional tagset vaguely resembling the popular *MULTEXT-East*[3] tagset utilized for several Slavic languages.

Language models for both systems, however, have been trained on the same source data – the 1.2 M token *Manually morphologically annotated corpus*[4] and the *SNC Morphology database*[5] covering approx. 100 K lemmas, yielding some 3.2 M inflected forms. This is why that, despite the fact that both systems do not produce exactly the same output, they are (almost) identical[6] in the amount of *OOV* items, that is rather high.

As both Slovak annotation systems explicitly indicate the *OOV* status of every token within a corpus, an analysis of the situation can be conveniently performed by the corpus manager, such as NoSketch Engine[7] [10]. In the *SNC* corpora, the *OOV* status is indicated by the "*XX*" value pf the "prec" attribute – this value can be observed in 54.5 million cases of 1.37 Gigatoken *prim-8.0-pubic-sane*[8] main corpus, which is 3.98% of all tokens.

In the web-based *Araneum Slovacum Maximum*[9], where the *OOV* state is indicated by the "*0*" value of the "ztag" attribute, the situation is even worse – 135.5 million *OOV*s out of 2.96 Gigatokens, i.e., 4.57%. This can be explained by the rather "low quality" of web data that, despite all efforts in cleaning and filtering the source texts, naturally contains lots of "noise" of different kinds.

3 The Task

The *OOV* lexical items observed in our corpora are of different nature. Besides the "true neologisms", i.e., words qualifying for inclusion even into the traditional dictionary, proper nouns (such as personal and geographical names) and their derivates, we can find also items traditionally not considered as "words" – various abbreviations, acronyms and symbols, URLs or e-mail addresses, parts of foreign language quotations and – above all – all sorts of "typos" and "errors". Inflected word forms apply to almost all previously mentioned categories, which makes the whole picture even more complex.

[1] We are aware of (at least) two more systems for morphosyntactic annotation of Slovak data that have been independently developed at Masaryk University in Brno and Charles University in Prague, respectively. These two systems, however, were not available for our work at the time of writing this paper.

[2] https://korpus.sk/morpho_en.html
[3] http://nl.ijs.si/ME/V4/
[4] https://korpus.sk/ver_r(2d)mak.html
[5] https://korpus.sk/morphology_database.html
[6] The differences are mainly caused by the fact that the *TreeTagger*-based system is also using word forms from the training corpus that were not present in the morphological database (mostly proper nouns) to ammend the morphological lexicon,
[7] https://nlp.fi.muni.cz/trac/noske
[8] https://korpus.sk/prim(2d)8(2e)0.html
[9] http://aranea.juls.savba.sk/aranea_about

In the following text we present an experiment aimed at amending the morphological lexicon used for training the language model(s) by a manually validated list of most frequent *OOV* items derived from an annotated web corpus. The annotation is to be performed by graduate students of foreign languages, in the framework of end-of-term assignment for the "Introduction to Corpus Linguistics" subject.

Having only limited "human power" (two groups with 46 students in total) at hand, we decided to follow the minimal two-fold setup (i.e., each item to be annotated by only two independent annotators) and make the task as simple as possible. This is why the annotators were not expected to check all the morphological categories provided by the respective tags, and they were asked to decide only on two parameters – lemma and word class (part of speech).

4 The Data

In the first step, we used data from the *Araneum Slovacum Maximum 17.09* web corpus of approx. 3 Gigatokens that has been independently tagged both by the *SNC MorphoDiTa* and the *Aranea TreeTagger* pipelines, and subsequently merged into a single vertical file. Then, we converted the original *SNC* morphological tags to "PoS-only" tags and produced a frequency list of all lexical items indicated as *OOV* by both taggers. This list has been further filtered to exclude word forms contained in the *Czech* morphological lexicon[10]. After deleting the unused parameters, the resulting lists contained the frequency, word form, lemma assigned by the *SNC* guesser and PoS information derived from the tag assigned by *TreeTagger* (*aTag*, using the *AUT*[11] notation). This decision has been motivated by an observation that *TreeTagger* is typically more successful in assigning morphological categories for unknown words than *MorphoDiTa*.

As we naturally could expect to be able to process only the rather small part of the list, after some experimenting with various thresholds, we decided to pass into annotation only items appearing 50 or more times, yielding to 77,169 items. This meant that each annotator would process approximately 3,300 items.

The example of source data (after discarding the frequency information and adding a unique Id) is shown in Table 1.

We can observe several phenomena here. The same lexical item is in some cases tagged as "foreign", while as "noun" or "adjective" in the others, and lemma form as well as its capitalization is sometimes guessed correctly, while sometimes not. It can be also seen, that many table items will in fact have to be merged after correcting the annotation, producing less total of correct lines.

The overall task for the annotators was to produce correct data for all lines in the table. To minimize the number of necessary keystrokes and to keep track of the changes, the data have been further modified to contain two newly added columns – *Lemmb* used as a template for correcting the value for *Lemma* (it is expected that most modifications will occur at the end of the respective string only) and *bTag* (to be filled only in case of wrong PoS assignment).

[10] https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-1836

[11] http://aranea.juls.savba.sk/aranea_about/aut.html

Table 1. Source Data

Id	*Word*	*Lemma*	*aTag*
sk_11184	dvojťažiek	dvojťažka	Nn
sk_11185	dvojťažiek	dvojťažky	Nn
sk_11186	dvojťažka	dvojťažka	Nn
sk_11187	Dvojťažka	dvojťažka	Nn
sk_11188	Dvojťažka	Dvojťažka	Nn
sk_11189	Dvojťažka	dvojťažka	Yx
sk_11190	Dvojťažka	Dvojťažka	Yx
sk_11191	dvojťažkách	dvojťažke	Nn
sk_11192	dvojťažke	dvojťažka	Nn
sk_11193	dvojťažkou	dvojťažka	Nn
sk_11194	dvojťažku	dvojťažka	Nn
sk_11195	dvojťažky	dvojťažka	Nn
sk_11196	dvojťažky	dvojťažky	Av
sk_11197	dvojťažky	dvojťažky	Nn
sk_11198	dvojtisícovku	dvojtisícovka	Nn
sk_11199	dvojtlačidlo	dvojtlačidlo	Nn
sk_11200	dvojtraktovú	dvojtraktový	Aj
sk_11201	dvojumývadlom	dvojumývadlom	Nn
sk_11202	dvojumývadlom	dvojumývadlom	Yx
sk_11203	dvojzákrutovej	dvojzákrutovej	Aj
sk_11204	dvojzákrutovej	dvojzákrutovej	Yx
sk_11205	dvojzápasovú	dvojzápasový	Aj
sk_11206	dvojzónovú	dvojzónový	Aj
sk_11207	dvolezite	dvolezite	Nn
sk_11208	dvolezite	dvolezite	Yx
sk_11209	Dvonča	Dvonča	Nn
sk_11210	Dvonča	Dvonč	Nn
sk_11211	Dvončom	Dvonča	Nn
sk_11212	Dvončom	Dvonč	Nn

As has been already mentioned, each item (line of the table) has to be annotated by two independent annotators. We decided, however, not to split the data in a straightforward way, but to assign each alphabetical segment of the data to *three* annotators using a rule as follows: each triple of lines will be split into three tuples containing first and second, first and third and second and third lines, respectively. Moreover, the whole lot of data has been split to three parts, so that each annotator could get three different sections of the alphabet in his or her data.

By applying this fairly "sophisticated" assignment scheme, we expected to improve the overall uniformity and quality of the output, as well as to prevent "collaboration" among students, as no two assigned lots were identical.

An excerpt of the data from Table 1 assigned to a single annotator is shown in Table 2.

Table 2. Data to Annotate

Id	*Word*	*Lemma*	*Lemmb*	*bTag*	*aTag*
sk_11184	dvojťažiek	dvojťažka	dvojťažka		Nn
sk_11185	dvojťažiek	dvojťažky	dvojťažky		Nn
sk_11187	Dvojťažka	dvojťažka	dvojťažka		Nn
sk_11188	Dvojťažka	Dvojťažka	Dvojťažka		Nn
sk_11190	Dvojťažka	Dvojťažka	Dvojťažka		Yx
sk_11191	dvojťažkách	dvojťažke	dvojťažke		Nn
sk_11193	dvojťažkou	dvojťažka	dvojťažka		Nn
sk_11194	dvojťažku	dvojťažka	dvojťažka		Nn
sk_11196	dvojťažky	dvojťažky	dvojťažky		Av
sk_11197	dvojťažky	dvojťažky	dvojťažky		Nn

Note that the "missing" every third Id results from the assignment scheme.

5 The Crowd Annotation

The split data has been uploaded as excel spreadsheets to a shared Google disk and assigned randomly to the respective annotators. The task has been assigned in the middle of

the semester, after the students already got acquainted with the basic concepts of corpus morphosyntactic annotation and acquired the elementary querying skills.

The instructions for annotating the data were as follows.

(A) Only *Lemmb* and *bTag* columns may be modified.

(B) If both *Lemma* and *aTag* values are correct, nothing has to be done.

(C) If *aTag* value is wrong, the correct value should be inserted in bTag.

(D) If *Lemma* value is wrong, it should be corrected in Lemmb.

(E) If the word form is obvious typo (missing or superfluous letter, exchanged letters), or the word does not contain the necessary diacritics, the correct lemma marked by an asterisk should entered in *Lemmb*.

(F) If the correct word form cannot be reconstructed by simple editing operations, i.e., cannot be recognized (e.g., part of the word as a result of hyphenation), the value of *bTag* will be "*Er*" (error).

(G) If the word form is obvious foreign word, the value of *bTag* will be "*Yx*".

(H) It is not necessary to evaluate whether the word form is "literary" – words of "lower" registers (such as slang) also have "correct" lemmas.

The annotators were also instructed to check all "non-obvious" items by querying the corpus and analyzing the respective contexts. The initial training was performed during one teaching lesson in a computer lab, so that possibly all frequent problems could be explained.

6 First Results and Problems

Out of 46 students, 43 managed to complete the assignments in time. Table 3 shows an example of the correctly annotated data.

Table 3. Annotated Data

Id	*Word*	*Lemma*	*Lemmb*	*bTag*	*aTag*
sk_11184	dvojťažiek	dvojťažka	dvojťažka		Nn
sk_11185	dvojťažiek	dvojťažky	dvojťažka		Nn
sk_11187	Dvojťažka	dvojťažka	dvojťažka		Nn
sk_11188	Dvojťažka	Dvojťažka	dvojťažka		Nn
sk_11190	Dvojťažka	Dvojťažka	dvojťažka	Nn	Yx
sk_11191	dvojťažkách	dvojťažke	dvojťažka		Nn
sk_11193	dvojťažkou	dvojťažka	dvojťažka		Nn
sk_11194	dvojťažku	dvojťažka	dvojťažka		Nn
sk_11196	dvojťažky	dvojťažky	dvojťažka	Nn	Av
sk_11197	dvojťažky	dvojťažky	dvojťažka		Nn
sk_11199	dvojtlačidlo	dvojtlačidlo	dvojtlačidlo		Nn
sk_11200	dvojtraktovú	dvojtraktový	dvojtraktový		Aj
sk_11202	dvojumývadlom	dvojumývadlom	dvojumývadlo	Nn	Yx
sk_11203	dvojzákrutovej	dvojzákrutovej	dvojzákrutový		Aj
sk_11205	dvojzápasovú	dvojzápasový	dvojzápasový		Aj
sk_11206	dvojzónovú	dvojzónový	dvojzónový		Aj
sk_11208	dvolezite	dvolezite	dôležitý*	Aj	Yx
sk_11209	Dvonča	Dvonča	Dvonč		Nn
sk_11211	Dvončom	Dvonča	Dvonč		Nn
sk_11212	Dvončom	Dvonč	Dvonč		Nn

We can see that PoS information was corrected in four cases, lemma form in nine cases and its capitalization in two cases. One lexical item was marked as "error", as it lacked all diacritics and used nonstandard spelling.

The quick analysis, however, revealed that the annotation is much below the expected quality. We will discuss some of the issues. The basic statistics is shown in Table 4.

Table 4. Results of Annotation

	Count	**%**	**%**
Assigned lines	77,169	100.00	
Lines annotated at least once	76,413	99.02	
Lines annotated twice	60,048	77.81	100.00
Lines agreed on lemma	39,469	51.15	65.73
Lines agreed on lemma and PoS	33,371	43.24	55.57

The rather low values of the raw inter-annotator agreement suggests that the resulting data has to be analyzed thoroughly before the procedure can be used within a similar larger-scale annotation attempt in the future.

The quick analysis revealed some frequent issues – different treatment of (prototypically) proper names written in lowercase, assigning PoS information to symbols and foreign words, incoherent use of asterisks, etc. Some of these issues can be solved by an automated procedure but some will require more detailed instruction so that a correct annotation could be obtained.

After merging the duplicate "fully agreed" items from the previous table, 27,135 unique lines were obtained. Table 5 shows the word class distribution of the resulting data.

Table 5. Annotated Data PoS Distribution

PoS	**Count**	**%**
Nn	20,043	73.86
Aj	5174	19.07
Pn	46	0.17
Nm	27	0.10
Vb	464	1.71
Av	261	0.96
Pp	8	0.03
Cj	10	0.04
Ij	42	0.15
Pt	24	0.09
Ab	185	0.68
Xy	1	0.00
Yx	490	1.81
Er	343	1.26
?	17	0.06
	27,135	100.00

The values in the table basically follow our expectations: most unrecognized items belong to main content word classes – nouns and adjectives. Moreover, out of the 20,043 words tagged as nouns, 14,190 (70.80%) begin with uppercase letter, i.e., they are most likely proper nouns.

The rather low value of the "Er" class can be explained by the observation that errors, despite their being frequent, rarely behave "paradigmatically", i.e., a single correct word form can produce many *different* incorrect ones.

7 Conclusions and Further Work

There were several goals to be achieved by the annotation. Firstly, we would like to produce a validated list of most frequent neologisms to be included in the morphological lexicon; in this stage, we even do not expect to generate full paradigms for those lexical items. Secondly, we wanted to get the list of the most frequent typos and other types of errors that could also be used as a supplement to that lexicon, but also as source data for a future system for data normalization. And lastly, we also wanted to obtain a list of most frequent foreign lexical items appearing in Slovak corpus data.

Although the detailed analysis of the annotated data is yet to be performed, some conclusions can be seen already. They can be summarized as follows:

(1) To minimize the consequences of students' failed assignments, a three-fold setup would be probably better.

(2) The Annotation Guidelines must be as precise as possible, showing not only the typical problems and their solutions, but also the seemingly "easy" cases. One-page instruction, as it was in our case, is definitely not sufficient.

(3) The most common errors were associated with the treatment of proper nouns. An automatic procedure based on frequencies of lower/uppercased word forms would most likely perform better.

(4) The other common issue was the proper form of lemma for adjectives (it should be masculine and nominative singular). As the morphology of Slovak adjectives is fairly regular, a procedure to fix it automatically would be feasible.

(5) One of the fairy frequent PoS ambiguity in our data was the "Nn"/"Yx" (noun/foreign) case. The manually annotated data, however, show that the real number of "foreigns" is rather low, yet in introduces a lot of noise into the annotation process. It would therefore be reasonable to substitute all tags for "foreigns" with that of "nouns" in the future annotation.

In the near future, besides the new round of a similar annotation effort with an improved setup, we would like to combine its results with those obtained in the framework of the ensemble tagging experiment described in our other work [11].

Acknowledgment

This work has been, in part, funded by the Slovak KEGA and VEGA Grant Agencies, Project No. K-16-022-00, and 2/0017/17, respectively.

References

[1] E. Estellés-Arolas and F. González-Ladrón-de-Guevara. Towards an Integrated Crowdsourcing Definition, Journal of Information Science, 38 (2): 189–200, doi:10.1177/0165551512437638.

[2] M. Šimková and R. Garabík. Slovenský národný korpus (2002–2012): východiská, ciele a výsledky pre výskum a prax. In Jazykovedné štúdie XXXI. Rozvoj jazykových technológií a zdrojov na Slovensku a vo svete (10 rokov Slovenského národného korpusu). Ed. K. Gajdošová – A. Žáková. Bratislava: VEDA 2014, pp. 35–64.

[3] D. "johanka" Spoustová, J. Hajič, J. Raab and M. Spousta. Semi-Supervised Training for the Averaged Perceptron POS Tagger. In Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009), pp. 763–771, Athens, Greece, March. Association for Computational Linguistics.

[4] J. Straková, M. Straka and J. Hajič. Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. In Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 13–18, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[5] V. Benko. Aranea: Yet Another Family of (Comparable) Web Corpora. In P. Sojka, A. Horák, I. Kopeček and Karel Pala (Eds.): Text, Speech and Dialogue. 17th International Conference, TSD 2014, Brno, Czech Republic, September 8–12, 2014. Proceedings. LNCS 8655. Springer International Publishing Switzerland, 2014.

[6] V. Benko. Two Years of Aranea: Increasing Counts and Tuning the Pipeline. In Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2016). – Portorož : European Language Resources Association (ELRA), 2016, pp. 4245–4248. ISBN 978-2-9517408-9-1.

[7] H. Schmid. Probabilistic Part-of-Speech Tagging Using Decision Trees. Proceedings of International Conference on New Methods in Language Processing, Manchester. 1994.

[8] H. Schmid. Improvements in Part-of-Speech Tagging with an Application to German. Proceedings of the ACL SIGDAT-Workshop, Dublin. 1995.

[9] R. Garabík and M. Šimková. Slovak Morphosyntactic Tagset. In Journal of Language Modeling. Institute of Computer Science PAS, 2012, Vol. 0, No. 1, pp. 41–63.

[10] P. Rychlý. Manatee/Bonito – A Modular Corpus Manager. In 1st Workshop on Recent Advances in Slavonic Natural Language Processing. Brno: Masaryk University, 2007. pp. 65–70. ISBN 978-80-210-4471-5.

[11] V. Benko and R. Garabík. Ensemble Tagging Slovak Web Data. Accepted for presentation at the SlaviCorp 2018 Conference, Prague, 24–26 September, 2018. Unpublished.

S. Krajči (ed.): ITAT 2018 Proceedings, pp. 130–137
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Michal Novák

ITAT

A Study on Bilingually Informed Coreference Resolution

Michal Novák

Charles University, Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
Malostranské náměstí 25, CZ-11800 Prague 1
mnovak@ufal.mff.cuni.cz

Abstract: Coreference is a basic means to retain coherence of a text that likely exists in every language. However, languages may differ in how a coreference relation is manifested on the surface. A possible way how to measure the extent and nature of such differences is to build a coreference resolution system that operates on a parallel corpus and extracts information from both language sides of the corpus. In this work, we build such a bilingually informed coreference resolution system and apply it on Czech-English data. We compare its performance with the system that learns only from a single language. Our results show that the cross-lingual approach outperforms the monolingual one. They also suggest that a system for Czech can exploit the additional English information more effectively than the other way round. The work concludes with a detailed analysis that tries to reveal the reasons behind these results.

1 Introduction

Cross-lingual techniques are becoming still more and more popular. Even though they do not circumvent the task of Coreference Resolution (CR), the research is mostly limited to cross-lingual projection. Other cross-lingual techniques remain a largely unexplored area for this task.

One of the yet neglected cross-lingual techniques is called *bilingually informed resolution*. It is an approach, in which decisions in a particular task are made based on the information from bilingual parallel data. Parallel texts must be available when a method is trained, but also at test time, that is when a trained model is applied to new data. In real-world scenarios, the availability of parallel data at test time requires the technique to apply a machine translation service to acquire them (MT-based bilingually informed resolution).

Nevertheless, for limited purposes it may pay off to use human-translated parallel data instead (corpus-based bilingually informed resolution). If it outperforms the monolingual approach, it may be used in building automatically annotated parallel corpora. Such corpora with more reliable annotation could be useful for corpus-driven theoretical research.[1] Furthermore, it can be also used for automatic processing. For instance, improved resolution on big parallel data might be leveraged in a weakly supervised manner to boost the models trained in a monolingual way.

The present work is concerned with corpus-based bilingually informed CR on Czech-English texts. Specifically, it focuses on resolution of pronouns and zeros, as these are the coreferential expressions whose grammatical and functional properties differ considerably across the languages. For instance, whereas in English most of non-living objects are referred to with pronouns in neuter gender (e.g. "*it*", "*its*"), genders are distributed more evenly in Czech. Information on Czech genders thus may be useful to filter out English candidates that are highly improbable to be coreferential with the pronoun. By comparison of its performance with a monolingual approach and by thorough analysis of the results, our work aims at discovering the extent and nature of such differences.

The paper is structured as follows. After mentioning related work (Section 2), we introduce a coreference resolver (Section 3), both its monolingual and cross-lingual variants. Section 4 describes the dataset used in experiments in Section 5. Before we conclude, the results of experiments are thoroughly analyzed (Section 6).

2 Related Work

Building a bilingually informed CR system requires a parallel corpus with at least the target-language side annotated with coreference. Even these days very few such corpora exist, e.g. Prague Czech-English Dependency Treebank 2.0 Coref [14], ParCor 1.0 [9] and parts of OntoNotes 5.0 [19].

It is thus surprising that the peak of popularity for such approach was reached around ten years before these corpora had been published. Harabagiu and Maiorano [10] present an heuristics-based approach to CR. The set of heuristics is expanded by exploiting the transitivity property of coreferential chains in a bootstrapping fashion. Moreover, they expand the heuristics even more, following mention counterparts in translations of source English texts to Romanian with coreference annotation. Mitkov and Barbu [13] adjust a rule-based pronoun coreference resolution system to work on a parallel corpus. After providing a linguistic comparison of English and French pronouns and their behavior in discourse, the authors distill their findings into a set of cross-lingual rules to be integrated into the CR system. In evaluation, they observe im-

[1] In case a cross-lingual origin of the annotation does not matter.

provements in resolution accuracy of up to 5 percentage points compared to the monolingual approach.

As for more recent works, the authors of [5] address the task of overt pronoun resolution in Chinese. Among the others they propose an MT-based bilingually informed approach. A model is built on Chinese coreference, exploiting Chinese features. These are augmented with English features, extracted from the Chinese texts machine-translated to English. It allows for taking advantage of English nouns' gender and number lists, which according to authors correspond to the distribution of genders and numbers over Chinese nouns.

Experiments of Novák and Žabokrtský [17], the first ones using bilingually informed CR on Czech-English data, are most relevant to the present work. With the focus on English personal pronouns only, their best cross-lingual configuration managed to outperform the monolingual CR by one F-score point. Taking advantage of a more developed version of their CR system, we extend their work in several directions. First, we explore the potential of such approach for a wider range of English coreferential expressions. Next, we perform experiments in the opposite direction, i.e. Czech CR informed by English. And finally, we provide a very detailed analysis of the results unveiling the nature of the cross-lingual aid.

3 Coreference Resolution System

For coreference resolution we adopt a more developed version of the resolver utilized in [17]. This new version builds on the monolingual Treex CR system [15], and augments it with the cross-lingual extension presented in [17]. The difference between the current system and the system in [17] lies mostly in that it can target a wider range of expressions, it exploits a richer feature set and the pre-processing stage analyzing the text to the tectogrammatical representation is of higher quality. Instead of listing all the changes, we briefly introduce the monolingual (Section 3.1) and the cross-lingual component (Section 3.2) of Treex CR from the scratch.[2]

3.1 Monolingual Resolution

Treex CR operates on the *tectogrammatical layer*. It is a layer of deep syntax based on the theory of Functional Generative Description [20]. The tectogrammatical representation of a sentence is a dependency tree with rich linguistic features consisting of the content words only. Furthermore, some surface ellipses are restored at this layer. It includes anaphoric zeros (e.g. zero subjects in Czech, unexpressed arguments of non-finite clauses in both English and Czech) that are introduced in the tectogrammatical layer with a newly established node.

[2]Please refer to [15] for more details on the monolingual component of the system.

The tectogrammatical layer is also the place, where coreference relations should be annotated. It is technically represented as a link between two coreferential nodes:[3] *the anaphor* (the referring expression) and *the antecedent* (the referred expression).

Each input text must be first automatically pre-processed up to this level of linguistic annotation. The CR system based on supervised machine learning then takes advantage of the information available in the annotation.

Pre-processing. The input text must undergo an analysis producing a tectogrammatical representation of its sentences before coreference resolution is carried out. We use pipelines for analysis of Czech and English available in the Treex framework [18]. The analysis starts with a rule-based tokenization, morphological analysis and part-of-speech tagging (e.g. [21] for Czech), dependency parsing to surface trees (e.g. MST parser [12] for English) and named entity recognition [22]. In addition, the NADA tool [3] is applied to help distinguish referential and non-referential occurrences of the English pronoun "*it*".

Tectogrammatical trees are created by a transformation from the surface trees. All function words are made hidden, morpho-syntactic information is transferred and semantic roles are assigned to tectogrammatical nodes [4]. On the tectogrammatical layer, certain types of ellipsis can be restored. The automatic pre-processing focuses only on restoring nodes that might be anaphoric. Such nodes are added by heuristics based on syntactic structures. The restored nodes include Czech zero subjects and both Czech and English zeros in non-finite clauses, e.g. zero relative pronouns, unexpressed arguments in infinitives, past and present participles.

Model design. Treex CR models coreference in a way to be easily optimized by supervised learning. Particularly, we use logistic regression with stochastic gradient descend optimization implemented in the Vowpal Wabbit toolkit.[4] Design of the model employs multiple concepts that have proved to be useful and simple at the same time.

Given an anaphor and a set of antecedent candidates, *mention-ranking* models [6] are trained to score all the candidates at once. On the one hand a mention-ranking model is able to capture competition between the candidates, but on the other hand features describe solely the actual mentions, not the whole clusters built up to the moment. Antecedent candidates for an anaphor (both positive and negative) are selected from the context window of a predefined size.

No anaphor detection stage precedes the coreference resolution. Unless another measure was taken, it would lead to all occurrences of the pronoun "*it*" labeled as referential, for instance. Nevertheless, the model determines

[3]A mention is determined only by its head in tectogrammatics. No mention boundaries are specified. Therefore, it is sufficient for a coreference link to determine only two nodes, the mentions' head nodes.

[4]`https://github.com/JohnLangford/vowpal_wabbit/wiki`

whether the anaphor is referential jointly with selecting its antecedent. This is ensured by adding a dummy candidate representing solely the anaphor itself. By selecting this candidate, the model claims that the anaphor is in fact non-referential.

Diverse properties of various types of coreferential relations (e.g. different referential scopes of personal and relative pronouns) encouraged us to model individual anaphor types separately. A specialized model is build for (1) personal and possessive pronouns in 3rd person (and zero subjects in Czech), (2) reflexive pronouns, (3) relative pronouns, and (4) zeros in non-finite clauses. Treex CR runs them in a sequence.

Features. The pre-processing stage enriches a raw text with a substantial amount of linguistic information. Feature extraction stage then uses this material to yield *features* consumable by the learning method. Features are always related to at most two nodes – an anaphor candidate and an antecedent candidate.

The features can be divided into three categories. Firstly, location and distance features indicate positions of the anaphor and the antecedent candidate in a sentence and their mutual distance in terms of words, clauses and sentences. Secondly, a big group of features reflects (deep) morpho-syntactic aspects of the candidates. It includes the mention head's part-of-speech tag and morphological features (e.g. gender, number, person, case), (deep) syntax features (e.g. dependency relation, semantic role) as well as some features exploiting the structure of the syntactic tree. Many of the features are combined by concatenation or by agreement, i.e. indicating whether the anaphor's value agrees with antecedent's one. Finally, lexical features focus on lemmas of the mentions' heads and their parents. These are used directly or through the frequencies collected in a large data of Czech National Corpus [1] indexed in a list of noun-verb collocations. Furthermore, all hypernymous concepts of a mention are extracted as features from ontologies (e.g. WordNet [7]) and named entity labels are also employed.

3.2 Cross-lingual Extension

The extension enables bilingually informed CR. Like the monolingual CR, it addresses coreference in one target language at a time. However, instead of data in single language, it must be fed with parallel data in two languages. Both language sides (Czech and English in this case) of the data must be first pre-processed with the pipelines analyzing the texts up to the diagrammatically layer. Furthermore, to facilitate the access to important information in the other language, the pre-processing stage also seeks for alignment between tectogrammatical nodes. The bilingually informed approach then augments the monolingual features with those accessing the other side of the parallel data. Design of the model remains the same as for the monolingual approach.

Alignment. It is central for our cross-lingual approach to have the English and Czech texts aligned on the level of tectogrammatical nodes. The alignment is based on unsupervised word alignment performed by MGIZA++ [8] trained on the data from CzEng 1.0 [4], and projected to the tectogrammatical layer. Furthermore, it is augmented with a supervised method [17] addressing selected coreferential expressions, including potentially anaphoric zeros.

Features. Cross-lingual features describe the nodes aligned to the coreferential candidates in the target language – the anaphor candidate and the antecedent candidate. To collect such nodes, we follow the alignment links connected to these two candidates. For each of the nodes, we take at most one of its aligned counterparts. In this way, we obtain at most two nodes aligned to the pair of potentially coreferential nodes, for which we can extract cross-lingual features. If no aligned counterpart is found, no cross-lingual features are added.

We extract two sets of cross-lingual features:

- *aligned_all*: it consists of all the features contained in a monolingual set for a given aligned language;
- *aligned_coref*: it consists of a single indicator feature, assigning the true value only if the two aligned nodes belong to the same coreferential entity. This feature can be activated only if there exists a monolingual coreference resolver for the aligned language. We employ Treex CR and its monolingual models for English and Czech, but any CR system, even a rule-based one, could be used.

We do not manually construct features combining both language sides. Nevertheless, such features are formed automatically by the machine-learning tool Vowpal Wabbit.

4 Datasets

We employ Prague Czech-English Dependency Treebank 2.0 Coref [14, PCEDT 2.0 Coref] to train and test our CR systems. It is a Czech-English parallel corpus, consisting of almost 50k sentence pairs (more on its basic statistics is shown in the upper part of the Table 1). The English part of the treebank is based on texts from the Wall Street Journal collected for the Penn Treebank [11]. The Czech part was manually translated from English. All texts have been annotated at multiple layers of linguistic representation up to the tectogrammatical layer.

Although PCEDT 2.0 Coref has been extensively annotated by humans, we strip almost all manual annotations and replace it by the output of the pre-processing pipeline (see Sections 3.1 and 3.2). The only manually annotated information that we retain are the coreferential links.

We do not split the data into train and test sections. All the experiments are conducted using 10-fold cross-validation.

Mention type	Czech	English
Sentences	49,208	49,208
Tokens	1,151,150	1,173,766
Tecto. nodes	931,846	838,212
Mentions (total)	183,277	188,685
Personal pron.	3,038	14,887
Possessive pron.	3,777	9,186
Refl. poss. pron.	4,389	—
Reflexive pron.	1,272	484
Zero subject	16,875	—
Zero in nonfin. cl.	6,151	29,759
Relative pron.	15,198	8,170
Other	132,577	126,199

Table 1: Basic and coref. statistics of PCEDT 2.0 Coref.

Mention type	Czech		English	
	monoling	biling	monoling	biling
Personal	$^{63.84}_{61.24}$ 62.51	$^{67.82}_{64.38}$ 66.06	$^{76.34}_{71.37}$ 73.77	$^{78.57}_{72.64}$ 75.49
Possessive	$^{71.93}_{71.51}$ 71.72	$^{75.73}_{74.85}$ 75.29	$^{80.07}_{79.54}$ 79.81	$^{81.46}_{81.00}$ 81.23
Refl. poss.	$^{85.61}_{85.42}$ 85.52	$^{87.70}_{87.04}$ 87.36	—	—
Reflexive	$^{66.91}_{56.60}$ 61.33	$^{67.24}_{55.66}$ 60.90	$^{77.31}_{72.67}$ 74.92	$^{75.88}_{71.01}$ 73.37
Zero subj.	$^{73.18}_{55.46}$ 63.10	$^{78.88}_{57.64}$ 66.61	—	—
Zero nonfin.	$^{78.98}_{41.51}$ 54.42	$^{81.52}_{42.63}$ 55.98	$^{71.48}_{54.62}$ 61.92	$^{73.31}_{54.75}$ 62.68
Relative	$^{81.51}_{79.94}$ 80.72	$^{83.48}_{81.62}$ 82.54	$^{83.47}_{76.23}$ 79.69	$^{85.76}_{77.13}$ 81.21
Total	$^{76.83}_{65.17}$ 70.52	$^{80.27}_{67.09}$ 73.09	$^{75.93}_{65.26}$ 70.19	$^{77.85}_{65.95}$ 71.41

Table 2: Anaphora scores of monolingual and bilingually informed coreference resolution.

As mentioned in Section 3.1, our CR system consists of four models targeting different types of mentions as anaphors. In evaluation, we split the anaphor candidates to even finer categories, namely: (1) personal pronouns, (2) possessive pronouns, (3) reflexive possessive pronouns, (4) reflexive pronouns, all four types of pronouns in the 3rd or ambiguous person, (5) zero subjects, (6) zeros in non-finite clauses, and (7) relative pronouns (the statistics of coreferential mentions is collected in the bottom part of Table 1). Driven by the findings in an analysis of Czech-English correspondences [16], these expressions are very interesting from a cross-lingual point of view, as they often transform to a different type or carry different grammatical properties, when translated. We assume this aspect is not so significant in case of nominal groups, for instance, which represent the majority of remaining mentions. The other types grouped under the category Other are demonstrative pronouns, pronouns in 1st and 2nd person etc. This category of anaphors is not targeted by our CR method.

5 Experiments

The following experiments compare the performance of the monolingual and bilingually informed system. Both systems are trained on the PCEDT dataset. All the design choices (except for the feature sets) and hyperparameter values are shared by both systems.

Evaluation measure. We expect different mention types to behave differently in the cross-lingual approach. Standard evaluation metrics (e.g. MUC [23], B^3 [2]), however, do not allow for scoring only a subset of mentions. Instead, we use the *anaphora score*, an anaphor-decomposable measure proposed by [15]. The score consists of three components: precision, recall, and F-score as a harmonic mean of the previous two. While precision expresses the success rate of a system averaged over all mentions labeled by the system as anaphoric, recall averages over all true anaphoric mentions. A decision on an anaphor candidate is correct if the system correctly labels it as non-anaphoric or the antecedent found by the system really belongs to the same entity as the anaphor. In the following tables, we use $^{\mathrm{P}}_{\mathrm{R}}$F to format the three components of the anaphora score.

Bilingually informed vs. Monolingual CR. Table 2 lists the anaphora scores measured on the output of 10-fold cross-validation. In overall, cross-lingual models succeed in exploiting additional knowledge from parallel data and perform better than the monolingual approach. The F-score improvement benefits mainly from a rise in precision, but recall also gets improved. In both languages, personal and possessive pronouns are the types that exhibit the greatest improvement. In Czech, the top-scoring mention types include zero subjects, too. Nevertheless, English as an aligned language seems to have a stronger impact on resolution in Czech (the difference between the systems is 2.5 F-score points) than Czech has on resolution in English (the difference of 1.2 F-score points).

6 Analysis of the Results

The results of experiments undoubtedly show the superiority of the cross-lingual CR over the monolingual one. Here, we delve more into the comparison of these two approaches. Firstly, we conduct a quantitative analysis of resolvers' decisions. It should show how many decision changes the switch to the cross-lingual approach introduces for individual mention types and what is the role of anaphoricity in these changes. Secondly, we inspect randomly sampled examples in a qualitative analysis. We attempt to disclose what are the typical examples when the system benefits from the other language and, on the other hand, if there is a systematic case when the cross-lingual approach hurts.

Mention type	Anaph				Non-anaph			
	Both ✓	Both ×	M > C	M < C	Both ✓	Both ×	M > C	M < C
Personal pron.	55.99	26.96	5.05	8.34	1.15	2.08	0.13	0.32
Possessive pron.	66.51	20.09	4.47	7.75	0.03	1.05	0.03	0.08
Refl. poss. pron.	82.45	9.59	2.64	4.27	0.11	0.89		0.05
Reflexive pron.	36.21	13.54	3.70	2.93	28.75	10.39	1.88	2.60
Zero subject	34.12	13.44	2.79	4.29	34.16	5.22	1.12	4.86
Zero in nonfin. cl.	68.54	12.62	2.94	5.24	3.82	6.08	0.42	0.32
Relative pron.	70.13	13.12	2.59	4.22	8.20	1.40	0.17	0.18
Total	53.76	14.20	3.00	4.73	17.96	3.52	0.61	2.22

Table 3: Comparison of resolution by the monolingual and the cross-lingual CR in Czech (M = Monolingual, C = Cross-lingual). The numbers are ratios (in %) of decision categories to which an anaphor candidate may fall.

6.1 Quantitative Analysis

Let us start with a quantitative analysis of improvements and worsenings with respect to anaphoricity and type of the anaphor candidate. Tables 3 and 4 show for Czech and English, respectively, how often the cross-lingual system (denoted as C) is better than the monolingual (denoted as M). Each anaphor candidate falls to one of the four categories based on how C and M decided on the candidate:

- both decisions were the same and correct (Both ✓),
- both decisions were the same but incorrect (Both ×),
- M's decision was correct while C's decision was incorrect (M > C),
- C's decision was correct while M's decision was incorrect (M < C).

A decision is either assignment of the anaphor candidate to a coreferential entity[5] or labeling it as non-anaphoric. The tables also distinguish if the candidate is in fact anaphoric or non-anaphoric. Numbers in the tables represent proportions (in %) of these categories aggregated over all instances. Every row thus sums to 100%.

Conditioning on anaphoricity allows us to directly relate this analysis to the anaphora scores shown in Table 2. Note that while resolution on anaphoric mentions may have an effect on both the precision and the recall component of the anaphora score, resolution on non-anaphoric mentions affects only the precision.

Changed decisions account for around 10% in both Czech and English. More importantly, whereas we see over 7% of decisions changed positively in Czech, it corresponds to 5.5% of decisions in English. This accords with the extents of improvement observed on anaphora score. In Czech, a difference between improved and worsened decisions is only a bit higher for anaphoric mentions. It means that the positive effect of English on resolution of Czech anaphoric mentions is about on par with its effect on resolution on non-anaphoric mentions. But conversely, Czech helps more in resolution of non-anaphoric mentions.

Let us zoom in to the individual mention types. The highest proportion of changed decisions appears for personal pronouns and zero subjects in Czech (14% instances) and for reflexive pronouns in English (12%). Interestingly, its effect on anaphora score cannot be more different. Czech personal pronouns and zero subjects are the mention types where the cross-lingual approach improves the anaphora score the most. On the other hand, English reflexive pronouns are the only mention type for which the resolution deteriorates with cross-lingual features. The systems' decisions differ the least for Czech reflexive possessive (7%) and English relative pronouns (6%). Here, we also observe a various effect on anaphora score. While the resolution of Czech reflexive possessives is hardly improved by English features, the small amount of changed decisions on English relative pronouns suffices to achieve one of the biggest improvements among English coreferential expressions.

Anaphora scores in Table 2 have already shown that basic reflexive pronouns are the only mention type, where the cross-lingual approach falls behind the monolingual one. The quantitative analysis of changed decisions confirms it, especially for anaphoric occurrences.

The gains of the Czech cross-lingual system on non-anaphoric mentions can be attributed mostly to zeros. Also thanks to the resolution on non-anaphoric mentions, the highest margin between the proportion of improved and worsened instances (5%) is observed on Czech zero subjects. It leads to one of the biggest improvement in terms of the anaphora F-score (see Table 2).

6.2 Qualitative Analysis

In the following, we scrutinize more closely what are the typical cases, where the cross-lingual system makes a different decision.

[5] Some of the anaphors that were assigned to the same entity (columns Both ✓ and Both ×) may have been in fact paired with different antecedents by each of the CR algorithms. As our anaphora score is agnostic to such changes, we do not distinguish such cases.

Mention type	Anaph				Non-anaph			
	Both ✓	Both ×	M > C	M < C	Both ✓	Both ×	M > C	M < C
Personal pron.	61.57	21.97	3.12	4.02	5.60	2.35	0.49	0.88
Possessive pron.	76.17	15.65	3.14	4.49	0.01	0.51	0.01	0.01
Reflexive pron.	69.78	15.00	7.17	5.22		2.83		
Zero in nonfin. cl.	44.10	16.74	3.82	3.83	16.55	11.08	1.26	2.61
Relative pron.	58.06	10.46	2.12	2.94	23.53	1.82	0.26	0.80
Total	54.46	16.87	3.35	3.84	12.81	6.31	0.77	1.60

Table 4: Comparison of resolution by the monolingual and the cross-lingual CR in English (M = Monolingual, C = Cross-lingual). The numbers are ratios (in %) of decision categories to which an anaphor candidate may fall.

Let us start with a motivating example. Results in Table 2 show that improvement of the bilingually informed system on Czech personal and possessive pronouns and zero subjects is twice as high than on their English equivalents. This observation genuinely surprised us. We had expected the opposite. Our supposition was based on the fact that Czech grammatical gender is more evenly distributed over nouns. We assumed Czech gender could help filtering out the English antecedent candidates whose Czech counterparts do not match the pronoun's counterpart. Although this still may be true, obviously, there are even stronger factors that operate in the opposite direction – from English to Czech.

Czech personal and possessive pronouns are the mention types that considerably benefit from the cross-lingual approach. Gender of the corresponding English pronoun appears to play an absolutely decisive role. Many times, gender of the Czech pronoun is masculine or feminine while gender of the English pronoun is neuter, as it is in Example 1. English pronoun's gender thus serves rather as an animacy feature, which cannot be reconstructed solely from the Czech pronoun. The correct antecedent is sometimes selected also with a help from the English pronoun's number.

(1) *Oponenti*$_{\text{m.pl}}$ *soudce*$_{\text{m.sg}}$ *Borka*$_{\text{m.sg}}$ *zvolili* ***bojiště***$_{\text{n.sg}}$ *drželi* ***ho***$_{\text{mn.sg}}$
opponents of judge Bork chose the battlefield held it

Oponenti soudce Borka zvolili bojiště, drželi ho a udrželi si ho.

Mr. Bork's opponents chose the battlefield, held it and kept it.

The analysis also shows that English syntax, which is more strict and thus easier to reconstruct, often helps in determining the correct antecedent. Example 2 shows the case, where neither English gender nor number could affect the resolver's decision. The correct decision is rather a result of a clear structure, where the syntactic objects in coordinated clauses very likely refer to the same entity.

(2) *kdo* *posbíral* ***plány***$_{\text{m.p}}$ *skupin*$_{\text{f.p}}$ *a* *sesmolil* ***je***$_{\text{mfn.p}}$ *do* *iniciativy*
who collected plans from groups and cobbled them into an initiative

Van de Kamp je ten, kdo posbíral plány různých radikálních ekologických skupin a sesmolil je do jedné neohrabané iniciativy…

Mr. Van de Kamp is the one who collected the plans from the various radical environmental groups and cobbled them into a single unwieldy initiative…

Some of the possessive pronouns benefit from another syntax-related factor. Example 3 shows the case where the correct decision was very likely affected by the fact that the aligned English possessive pronoun ("*its* Opel line") is in a short context preceded by a construction with a possessive adjective ("*GM's* interest"). Not only the possessed objects does not have to be the same, but the possessivity factor also suppresses the unclear gender agreement in Czech ("*jeho* /its/" can be of masculine or neuter gender, whereas "*společnost* /company/" is of feminine gender and the gender of "*GM*" may be arbitrary).

(3) *zájem*$_{\text{m.s}}$ ***společnosti GM***$_{\text{fm.s}}$ *o* *společnost Jaguar*$_{\text{fm.s}}$
interest GM-company's in Jaguar company
odráží *touhu*$_{\text{f.s}}$ *pomoci*$_{\text{f.s}}$ *zpestřit* *produkty*$_{\text{m.p}}$
reflects a desire to help diversify products
této společnosti$_{\text{f.s}}$ *na* *trhu*$_{\text{m.s}}$ *s* *vozy*$_{\text{m.p}}$. ***jeho***$_{\text{mn.s}}$
of this company in market with cars . its
série *Opel*
line Opel

Zájem společnosti GM o společnost Jaguar odráží touhu pomoci zpestřit produkty této americké společnosti na rostoucím trhu s luxusními vozy. Jeho série Opel má zavedený image…

GM's interest in Jaguar reflects a desire to help diversify the U.S. company's products in the growing luxury-car segment of the market. Its Opel line has a solid image…

Zero subjects is another Czech mention type for which a large improvement of the cross-lingual approach is observed. Anaphoric zero subjects benefit from the aspects similar to those we mentioned for personal pronouns: gender and number of the anaphor, more strict syntactic constraints in English etc. English gender may be even more important here, as the gender of a subject zero is impossible to be recognized just from the form of the governing verb, if the verb is in present tense.

While inspecting a sample of changed decisions for English personal and possessive pronouns, we do not witness many examples of clear influence by Czech gender or number. As for the personal pronouns, influence of gender or number is most often combined with the pure fact that the English pronoun has an aligned counterpart in Czech. For many of such pronouns, the option that the pronoun is non-anaphoric can then be discarded. The strength of this

aspect very likely accounts for the fact that the majority of most confident decision changes were in fact labeled as non-anaphoric by the monolingual system (e.g. in Example 4). Czech language side of the data thus help correctly label these pronouns as anaphoric.

(4) *Compelled* ***service*** *is unconstitutional* ***It*** *is also unwise*
Nucená služba$_{f,s}$ je protiústavní $\varnothing_{f,s}$ Je také nerozumná
Compelled service is unconstitutional. It is also unwise and unenforceable.
Nucená služba je protiústavní. Je také nerozumná a nevynutitelná.

Similarly, most of the improvements among English possessive pronouns do not result from additional information on gender and number from Czech. The cross-lingual system rather takes advantage of the cases where a reflexive possessive pronoun is a Czech counterpart of the English possessive pronoun (see Example 5), or the cases where the pronoun has no Czech counterpart at all. In all these cases, the syntactic subject of the clause in which the pronoun lies is a preferred antecedent.

(5) ***Digital Equipment Corp.*** *announced* ***its*** *line of computers*
společnost Digital Equipment Corp. představila svou řadu počítačů
The hottest rivalry in the computer industry intensified sharply yesterday as Digital Equipment Corp. announced its first line of mainframe computers…
Nejžhavější rivalita v počítačovém průmyslu se včera notně přiostřila, když společnost Digital Equipment Corp. představila svou první řadu centrálních počítačů…

Back to the Czech zero subjects. Many of these expressions reconstructed during the automatic analysis are in fact superfluous. It is usually a consequence of a parsing error, when the real subject of a clause is not recognized (e.g. the word "*společnosti* /companies/" in Example 6). This error subsequently propagates to a wrong decision of the monolingual resolver (the word "*zpráva* /report/" labeled as an antecedent). Any superfluous zero subject may be correctly resolved in two ways: (1) labeling it as non-anaphoric, or (2) linking it to the expression that plays the same role in the sentence. We observe that 85% of the decisions corrected by the cross-lingual system are fixed in the former way. And a missing English counterpart of the superfluous zero plays a significant role in such decisions.

(6) *Avšak* ~~***zpráva***~~ *uvádí že společnosti*$_{\text{subj}}$ $\varnothing_{\text{subj}}$
But the report said that companies –
platí více daní
are paying more taxes
Avšak zpráva uvádí, že ačkoliv společnosti platí více daní, mnoho jich stále platí méně, než činí zákonná sazba.
But even though companies are paying more taxes, many are still paying less than the statutory rate, the report said.

In a similar way, detection of English non-anaphoric zeros in non-finite clauses can be boosted by Czech features. If the zero is non-anaphoric, its governing clause usually remains non-finite in Czech or it turns into a noun phrase. For instance, in Example 7 the entity which performs the act of "*hiring*" is not specified in the context of a given sentence, which is emphasized by the use of the noun "*nábor*" as a Czech translation of the participle. The automatically parsed structure of such cases is the same: since Czech non-subject zeros are rarely reconstructed by Treex linguistic pre-processing, there is usually no counterpart for the English zero to align with.

(7) ~~***Fear***~~ *of AIDS hinders* $\varnothing_{\text{actor}}$ *hiring*
Strach z AIDS komplikuje – nábor$_{noun}$
Fear of AIDS hinders hiring at few hospitals.
Strach z AIDS komplikuje nábor v několika nemocnicích.

The category of relative pronouns specified in terms of automatically set attributes may contain lots of pronouns that are in fact interrogative or fused. Such instances account for the majority of non-anaphoric English relative pronouns, correctly discovered by the cross-lingual system but not by the monolingual one.

Finally, we sought for the reasons of worsenings within a category of Czech and English reflexive pronouns. The worst decisions made by the cross-lingual method in Czech are on the pronouns that ended up resolved as non-anaphoric. Most of the time these incorrectly labeled pronouns have no alignment to English, thus no cross-lingual features related to the anaphor can be activated. On the other hand, the English cross-lingual resolver makes the most serious mistakes by selecting a wrong antecedent. In these cases, the pronouns are most often aligned to their Czech counterparts and these counterparts are actually often correct. Yet, the choice of the English antecedent seems to be random, regardless whether the Czech counterpart is labeled as coreferential with its correct antecedent, or the counterpart is any of the words *sám* or *samotný*, which should indicate emphatic use of the English reflexive pronoun.

7 Conclusion

This work conducts experiments on bilingually informed coreference resolution on Czech-English data. Comparing this cross-lingual approach to a monolingual resolver, we discovered that English helps more in resolution of Czech expressions than vice versa. A quantitative analysis shows that while English facilitates resolution of both Czech anaphoric and non-anaphoric mentions, Czech primarily helps to identify non-anaphoric mentions. The qualitative analysis reveals main reasons for improvements and worsenings all over the mention types. The most surprising finding is that the information on English gender seems to be improving resolution of Czech coreference more than vice versa. The animacy feature hidden in English gender appeared to be stronger than more even distribution of Czech genders across nouns.

Acknowledgments

This project has been funded by the Czech Science Foundation grant GA-16-05394S. This work has been also sup-

ported and has been using language resources developed and/or stored and/or distributed by the LINDAT/CLARIN project No. LM2015071 of the Ministry of Education, Youth and Sports of the Czech Republic.

References

[1] *Czech National Corpus – SYN2005*. Institute of Czech National Corpus, Faculty of Arts, Charles University, Prague, Czech Republic, 2005.

[2] A. Bagga and B. Baldwin. Algorithms for Scoring Coreference Chains. In *In The First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*, pages 563–566, 1998.

[3] S. Bergsma and D. Yarowsky. NADA: A Robust System for Non-referential Pronoun Detection. In *Proceedings of the 8th International Conference on Anaphora Processing and Applications*, pages 12–23, Berlin, Heidelberg, 2011. Springer-Verlag.

[4] O. Bojar, Z. Žabokrtský, O. Dušek, P. Galuščáková, M. Majliš, D. Mareček, J. Maršík, M. Novák, M. Popel, and A. Tamchyna. The Joy of Parallelism with CzEng 1.0. In *Proceedings of LREC 2012*, Istanbul, Turkey, May 2012. ELRA, European Language Resources Association.

[5] C. Chen and V. Ng. Chinese Overt Pronoun Resolution: A Bilingual Approach. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1615–1621, Québec City, Québec, Canada, 2014. AAAI Press.

[6] P. Denis and J. Baldridge. A Ranking Approach to Pronoun Resolution. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, pages 1588–1593, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[7] C. Fellbaum. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press, 1998.

[8] Q. Gao and S. Vogel. Parallel Implementations of Word Alignment Tool. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, pages 49–57, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[9] L. Guillou, C. Hardmeier, A. Smith, J. Tiedemann, and B. Webber. ParCor 1.0: A Parallel Pronoun-Coreference Corpus to Support Statistical MT. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC-2014)*, pages 3191–3198, Reykjavik, Iceland, 2014. European Language Resources Association (ELRA).

[10] S. M. Harabagiu and S. J. Maiorano. Multilingual Coreference Resolution. In *Proceedings of the Sixth Conference on Applied Natural Language Processing*, pages 142–149, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[11] M. Marcus, B. Santorini, M. A. Marcinkiewicz, and A. Taylor. Penn Treebank 3, 1999.

[12] R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. Non-projective Dependency Parsing Using Spanning Tree Algorithms. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[13] R. Mitkov and C. Barbu. Using Bilingual Corpora to Improve Pronoun Resolution. *Languages in contrast*, 4(2), 2003.

[14] A. Nedoluzhko, M. Novák, S. Cinková, M. Mikulová, and J. Mírovský. Coreference in Prague Czech-English Dependency Treebank. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, pages 169–176, Paris, France, 2016. European Language Resources Association.

[15] M. Novák. Coreference Resolution System Not Only for Czech. In *Proceedings of the 17th conference ITAT 2017: Slovenskočeský NLP workshop (SloNLP 2017)*, volume 1885 of *CEUR Workshop Proceedings*, pages 193–200, Praha, Czechia, 2017. CreateSpace Independent Publishing Platform.

[16] M. Novák and A. Nedoluzhko. Correspondences between Czech and English Coreferential Expressions. *Discours: Revue de linguistique, psycholinguistique et informatique.*, 16:1–41, 2015.

[17] M. Novák and Z. Žabokrtský. Cross-lingual Coreference Resolution of Pronouns. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 14–24, Dublin, Ireland, 2014. Dublin City University and Association for Computational Linguistics.

[18] M. Popel and Z. Žabokrtský. TectoMT: Modular NLP Framework. In *Proceedings of the 7th International Conference on Advances in Natural Language Processing*, pages 293–304, Berlin, Heidelberg, 2010. Springer-Verlag.

[19] S. Pradhan, A. Moschitti, N. Xue, H. T. Ng, A. Björkelund, O. Uryupina, Y. Zhang, and Z. Zhong. Towards Robust Linguistic Analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria, 2013. Association for Computational Linguistics.

[20] P. Sgall, E. Hajičová, and J. Panevová. *The Meaning of the Sentence in Its Semantic and Pragmatic Aspects*. D. Reidel Publishing Company, Dordrecht, Netherlands, 1986.

[21] J. Straková, M. Straka, and J. Hajič. Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 13–18, Baltimore, Maryland, 2014. Association for Computational Linguistics.

[22] J. Straková, M. Straka, and J. Hajič. Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 13–18, Baltimore, Maryland, 2014. Association for Computational Linguistics.

[23] M. Vilain, J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman. A Model-theoretic Coreference Scoring Scheme. In *Proceedings of the 6th Conference on Message Understanding*, pages 45–52, Stroudsburg, PA, USA, 1995. Association for Computational Linguistics.

S. Krajči (ed.): ITAT 2018 Proceedings, pp. 138–143
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Jindřich Libovický, Rudolf Rosa, Jindřich Helcl, and Martin Popel

ITAT

Solving Three Czech NLP Tasks End-to-End with Neural Models

Jindřich Libovický, Rudolf Rosa, Jindřich Helcl, and Martin Popel

Institute of Formal and Applied Linguistics, Faculty of Mathematics and Physics, Charles University,
Malostranské náměstí 25, 118 00 Praha, Czech republic
surname@ufal.mff.cuni.cz

Abstract: In this work, we focus on three different NLP tasks: image captioning, machine translation, and sentiment analysis. We reimplement successful approaches of other authors and adapt them to the Czech language. We provide end-to-end architectures that achieve state-of-the-art results on all of the tasks within a single sequence learning toolkit. The trained models are available both for download as well as in an online demo.

1 End-to-End Training

Traditionally, solving tasks such as machine translation or sentiment analysis required complex processing pipelines consisting of tools which transformed one explicit representation of the data into another, with the structure of the internal representations defined by the system designer. In machine translation, we would devise explicit word alignment links, extract phrase tables, train a language model, etc.; in sentiment analysis, we could label the data with part-of-speech tags, decode their syntactic structure, and/or assign them with semantic labels. All of these more-or-less linguistically motivated internal representations are not inherently required to produce the desired output, but have been devised as clever and useful ways to break down the large and hard task into smaller and manageable substeps.

With the advent of end-to-end training of deep neural networks (DNN), the need for most of this has been eliminated. In the end-to-end learning paradigm, there is only one model, directly trained to produce the desired outputs from the inputs, without any explicit intermediate representations. The system designer now only has to design a rather generic architecture of the system. It mostly does not enforce any complex explicit representations and processing steps, but rather offers opportunities for the DNN to devise its own notion of intermediate representations and processing steps through training.

This also means that similar architectures can be used to solve very different tasks. Rather than by the nature of the task itself, the structure of the DNN to use is mostly determined by the structure of the input and output – e.g. image inputs are processed by two-dimensional convolutions, while text inputs are processed by one-dimensional convolutions, recurrent units, and/or attentions; classification can produce its output in one step, while text generation is better done iteratively using recurrent decoders; etc.

Thanks to that, a single general framework can be used to solve many different tasks. One just needs to transform the inputs and outputs into a suitable format, define an adequate network structure, and let the system train for a few weeks.

Sadly, the burden of hyperparameter tuning has not been alleviated by DNNs, but rather made worse by the computational costliness of the training. However, with a bit of experience, one is often able to propose a suitable architecture and hyperparameter values at the first attempt, already achieving very competitive results even without any further tuning.

1.1 Our contribution

Most of the papers in the field only evaluate their setups on English datasets. In our work, we try to rectify this shortcoming by reimplementing existing state-of-the-art approaches in the Neural Monkey framework [14] and training them on existing Czech datasets.

Neural Monkey is an open-source toolkit for sequence-to-sequence learning, implemented in the TensorFlow library [1]. The toolkit is designed to be easily extensible in order to support fast prototyping of architectures for various NLP tasks. It is freely available on GitHub[1] under the BSD license, allowing both non-commercial and commercial use of the toolkit.

We decided to focus on three rather varied tasks – sentiment analysis, machine translation, and image captioning. For each of the tasks, we reimplemented one or more existing state-of-the-art architectures within Neural Monkey and trained it on available datasets. Our evaluations show that we manage to reach or surpass state-of-the-art results for all the three tasks.

As we wish to encourage other NLP researchers to focus on Czech language, we make sure that our source codes, our configuration files and our trained models are all freely available to anyone interested to use them, to study them and to build upon them. With our work, we hope to establish well-performing approaches for Czech NLP, as well as to allow e.g. investigation of the internals of the trained models to try to decipher in what ways language seems to be implicitly captured in them. Moreover, we have created a simple web-based demo that allows anyone to easily apply our models to any input data, intended to popularize deep learning and its applications for the Czech audience.

[1] https://github.com/ufal/neuralmonkey

2 Sentiment Analysis

The goal of the sentiment analysis tasks is to decide whether a text expresses positive, neutral or negative judgment on its topic, sometimes also a degree of the positivity or negativity.

2.1 Architecture

We use a state-of-the-art architecture by Lin et al. [20]. The architecture processes the text with a bi-directional LSTM network [15, 11]. After that the attention mechanism is applied several times, each time with a different trained query vector; this is usually referred to as the architecture featuring multiple attention heads. This gives us a set of context vector, each of them being a different weighted average of the LSTM states.

For English, Lin at el. [20] achieved new state-of-the-art results on the Yelp dataset[2] which contains texts of restaurant reviews and the number of stars the users assigned to the review. The goal of the prediction is an automatic assignment of the stars.

After replicating the results on the English dataset, we evaluated the same approach on a Czech dataset. We also experimented with architectures based on processing the input with a convolutional network or a recurrent network followed by max-pooling in time [16].

All models use embeddings of size 300 and a classifier with 100 hidden units. In the experiments with CNN, we used kernels of size 3, 4 and 5 with output dimension 100. The LSTM network used 300 hidden units in both directions. The self-attentive layer used 10 heads and a hidden layer of 300 dimensions.

2.2 Dataset

The largest existing Czech dataset for sentiment analysis is the *CSFD CZ* dataset [12], which is available online[3] under the CC-BY-NC-SA license. It consists of 91,379 movie reviews from ČSFD,[4] a Czechoslovak film database.

The textual reviews are on average 60 tokens long, and bear a rating of 0 to 6 stars, which the authors of the dataset mapped into three classes: negative (0-2 stars), neutral (3-4 stars), and positive (5-6 stars). The three classes are represented rather uniformly, each being assigned to 32%-34% reviews.

We split off 2,000 reviews for validation and another 2,000 for testing (there is no official split of the dataset), retaining the nearly uniform distribution of the classes, as well as other characteristics of the dataset such as average review length.

We use tokenization from the Moses MT toolkit [18] and post-process the tokenization in order to normalize emoticons and repetitive vowels that are often used for emphasis. We use vocabulary of 50k tokens appearing at least 5 times in the training data.

[2] https://www.yelp.com/dataset/

[3] http://liks.fav.zcu.cz/sentiment/

[4] https://www.csfd.cz/

Setup	Accuracy
Most frequent class	35.70 %
Maxpool on embeddings	$80.3 \pm .1$ %
CNN + maxpool	$79.2 \pm .1$ %
SAN on embeddings	$80.1 \pm .1$ %
SAN on LSTM	$80.8 \pm .1$ %
Lenc+ [19]	71.00 %
Brychcín+ [7]	81.53 %

Table 1: Quantitative evaluation of sentiment analysis

2.3 Evaluation

The evaluation in Table 1 show that no matter which particular architecture we use, we achieve accuracies around 81 %. We hypothesize that this already approaches the highest accuracy practically achievable on the dataset, and that all of the model architectures are sufficiently powerful to achieve this accuracy.

Similarly to our approach, Lenc and Hercig [19] experiment with convolutional networks and max-pooling [16], however due to a small vocabulary and limited input length, they report scores which are ten percentage points smaller than ours.

To the best of our knowledge, the best result on this dataset has been reported for the "ME + sspace + Dir" setup of Brychcín+ [7]; they report a ± 0.3 confidence interval for their accuracy, which our best result also falls into. The authors use a complex setup combining a Maximum Entropy classifier with an unsupervised extension that incorporates global context into the classification, based on the assumption that reviews for the same target (movie) tend to bear similar labels; this extension brings them approximately +3 accuracy points. We do not incorporate this mechanism into our setup; in fact, our system does not use the information about the identity of the movie at all. This shows that our model is stronger in a restricted variant of the task – predicting the sentiment solely from the plain text.

3 Machine Translation

Machine translation (MT) is one of the most well-studied problems from NLP. In general, the goal of MT is given a sentence in a source language, generate a sentence in a target language which as similar meaning as possible to the source sentence.

3.1 Architecture

We use our implementation of the self-attentive architecture called the Tranformer [27]. Our implementation

is compatible with the official implementation in Tensor2Tensor [26] and we can thus take advantage of highly optimized training procedure.

The architecture uses the encoder-decoder scheme [3]. Unlike the original sequence-to-sequence models which were based on recurrent neural networks, the Transformer model uses a stack of self-attentive and feed-forward layers.

In the self-attentive layers, we use the state as a query to an attention over the remaining states of the layer and output a weighted combination of the states. This is always followed by a feed-forward layer. All layers are normalized [2] and interconnected with residual connection [13] to ensure better gradient flow during training.

The decoder uses also attention to the encoder after each self-attentive layer. The decoder is autoregressive. In every time step, a new word is generated and the stack of all the layers applied on the text generated so far, including the newly generated word.

We use hyper-parameters and training strategy proposed by Popel and Bojar [23] who train the model in Tensor2Tensor. A vocabulary of 32,000 subwords is shared by the English encoder and Czech decoder. The network uses 16 self-attentive heads and a hidden layer of dimension 1,024. It is trained using the Adam optimizer [17] with the beta parameter set to 0.998 and the learning rate to 0.2 with 16,000 warmup steps, using a batch size of 1,500 and checkpoint averaging.

3.2 Dataset

We use CzEng 1.7[5] [5] Czech-English parallel corpus in a filtered version [23] which contains 57M pairs of parallel sentence pairs.

The model is validated on WMT13 test set and evaluated on WMT17 [6] test set from the news domain.

3.3 Evaluation

We evaluate the model on the WMT17 test set [6]. It is a test set that was used for system comparison in an annual competition in MT. Unlike the other 2 tasks which are rarely solved for Czech, English-to-Czech translation is annually evaluated within the WMT competition where it serves as an example of a highly inflected language.

The quantitative results are in Table 3, examples of the outputs in Table 2. As far as we know, this is the best publicly reported MT system for English-to-Czech translation.

Our best performing model was obtained by training for 8 days on 8 GPUs.

4 Image Captioning

In image captioning, the task is to provide a short textual description of a given image – i.e., the input for the task is an image (a two-dimensional matrix of bits, where each bit is represented by the values of its red, green and blue channel), and the output is a caption (a sequence of words).

As the Czech image captioning dataset is very new, we believe to be the first ones to train models for the image captioning task for Czech.

4.1 Architecture

We re-implement an attentive architecture by Xu et al. [29]. The model uses pre-trained convolutional map from networks for image classification on the ImageNet dataset [8], these are used as input to a RNN decoder with attention mechanism [3] originally introduced in context of MT.

Image features are extracted with Resnet50 v2 ($8 \times 8 \times 2048$) [13], captions are tokenized and truecased Moses style [18]. We use an RNN decoder [3] with conditional GRU [10] with dimensionality 1024, and our word embeddings have 500 dimensions. For Czech experiments, we use a vocabulary of 5,521 tokens, i.e., tokens that appear at least four times in the training data. For English, we use a vocabulary of 7,752 tokens appearing at least 5 times.

The model is optimized using the Adam optimizer [17] with default parameters and mini-batch size 64. Because we cannot rely on an extrinsic evaluation metric, we perform early stopping on reference captions perplexity.

At the inference time, we use a beam search of width 5 with length penalty 1.0 [28].

4.2 Dataset

We use a recently acquired Czech version of the Multi30k dataset [9] which contains translations of the originally English captions from the Flickr30k dataset [22].

The dataset uses 29,000 images for training, 1,014 for validation and 1,000 for testing. Unlike the original Flickr30k dataset which contains 5 independent descriptions for each image, we only have one Czech sentence for each image.

This means we can have only have one reference sentence for the evaluation which makes the evaluation less robust than in case of English.

4.3 Evaluation

Image captioning is usually evaluated using metrics originally developed for machine translation.

There is only one reference in the dataset, while the standard is to evaluate with BLEU [21] or METEOR [4] score against 6 references. In MT, 4 references are the standard [21], and 1 reference is typical in practice. In image captioning, the captions are quite short, and there is a much higher degree of freedom, which is why as many as 6 references are typically used. With only 1 reference available, BLEU cannot be reliably used here. However,

[5] http://ufal.mff.cuni.cz/czeng/czeng17

source:	The next chance won't come until winter.
system output:	Další příležitost přijde až v zimě.
reference:	Další šance přijde až v zimě.
source:	All private correspondence and images should remain private.
system output:	Veškerá soukromá korespondence a obrazy by měly zůstat soukromé.
reference:	Všechna soukromá korespondence a všechny soukromé obrázky by soukromé měly zůstat.

Table 2: An example of the outputs of the MT system.

cs output:	Skupina lidí stojí ve sněhu.
cs reference:	Skupina lidí stojící před iglú.
en output:	A group of people are standing in front of a building.
en reference:	A group of people wearing snowshoes, and dressed for winter hiking, is standing in front of a building that looks like it's made of blocks of ice. The people are quietly listening while the story of the ice cabin was explained to them. A group of people standing in front of an igloo. Several students waiting outside an igloo.

Figure 1: An example of an output of the image captioning system.

model	BLEU
Popel and Bojar [23] (ours)	23.8
WMT17 winner [25]	22.8
Google Translate [28]	20.8

Table 3: Qualitative evaluation of the MT model.

model	BLEU	METEOR	chrF3
Xu et al. [29]	19.1	18.5	—
ours (English)	19.7	17.0	0.17
ours (Czech)	2.3	7.2	0.14

Table 4: Quantitative results of the image captioning models.

as more references are not available, we use the chrF3 metric [24], which is based on character *n*-grams rather than word *n*-grams, and has thus a higher chance of providing at least somewhat useful evaluation scores (even though we note that they are still very unreliable).

We believe our work to be the first to perform image captioning in Czech language. As can be seen in Table 4, the standard evaluation shows rather low scores for Czech. However, when investigating the data, we found the produced image labels to be usually correct, even if rather simple and generic. See Figure 1 for an example of an input image together with its captions produced by our system.

5 Conclusion

We implemented and trained models for English-to-Czech machine translation, sentiment analysis of Czech texts, and image captioning in Czech within Neural Monkey, using approaches reported to be state-of-the-art for other languages (typically English). We gathered and standardized existing datasets, adapted the Neural Monkey toolkit where necessary, and trained and tuned the tools. Our evaluation shows that the resulting tools reach or surpass state-of-the-art for all three tasks. Both the source codes and the trained models are available online under free licences.[6][7] The tools are also available as an online demo.[8]

As a future work, we plan to add more tasks, especially text summarization.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang

[6] https://github.com/ufal/neuralmonkey
[7] http://hdl.handle.net/11234/1-2839
[8] https://ufal.mff.cuni.cz/grants/lsd

Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[4] Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

[5] Ondřej Bojar, Ondřej Dušek, Tom Kocmi, Jindřich Libovickỳ, Michal Novák, Martin Popel, Roman Sudarikov, and Dušan Variš. Czeng 1.6: enlarged Czech-English parallel corpus with processing tools dockered. In *International Conference on Text, Speech, and Dialogue*, pages 231–238. Springer, 2016.

[6] Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. Findings of the 2017 conference on machine translation (WMT17). In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 169–214, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[7] Tomáš Brychcín and Ivan Habernal. Unsupervised improving of sentiment analysis using global target context. In *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*, pages 122–128, Hissar, Bulgaria, September 2013. INCOMA Ltd. Shoumen, BULGARIA.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255, Miami, FL, USA, jun 2009. IEEE, IEEE Computer Society.

[9] Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. Multi30k: Multilingual English-German image descriptions. In *Proceedings of the 5th Workshop on Vision and Language*, pages 70–74. Association for Computational Linguistics, 2016.

[10] Orhan Firat and Kyunghyun Cho. Conditional gated recurrent unit with attention mechanism. https://github.com/nyu-dl/dl4mt-tutorial/blob/master/docs/cgru.pdf, May 2016. Published online, version `adbaeea`.

[11] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.

[12] Ivan Habernal, Tomáš Ptáček, and Josef Steinberger. Sentiment analysis in Czech social media using supervised machine learning. In *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 65–74, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, TODO, TODO 2016. IEEE Computer Society.

[14] Jindřich Helcl and Jindřich Libovický. Neural Monkey: An open-source tool for sequence learning. *The Prague Bulletin of Mathematical Linguistics*, (107):5–17, 2017.

[15] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[16] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics.

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[18] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June 2007.

[19] Ladislav Lenc and Tomáš Hercig. Neural networks for sentiment analysis in Czech. In *Proceedings of the 16th ITAT: Slovenskočeský NLP workshop (SloNLP 2016)*, pages 48–55, Bratislava, Slovakia.

[20] Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *CoRR*, abs/1703.03130, 2017.

[21] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[22] Bryan A Plummer, Liwei Wang, Chris M Cervantes, Juan C Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pages 2641–2649. IEEE, 2015.

[23] Martin Popel and Ondřej Bojar. Training tips for the Transformer model. *The Prague Bulletin of Mathematical Linguistics*, 110:43–70, April 2018.

[24] Maja Popović. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[25] Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Ger-

mann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams. The University of Edinburgh's neural MT systems for WMT17. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 389–399, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[26] Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. Tensor2tensor for neural machine translation. *CoRR*, abs/1803.07416, 2018.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 6000–6010, Long Beach, CA, USA, December 2017. Curran Associates, Inc.

[28] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.

[29] Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 2048–2057. JMLR.org, 2015.

“W3-ChaT” — WWW — Challenges and Trends (W3-ChaT 2018)

The W3-ChaT workshop is focused on addressing existing challenges we are facing today on the Web (and in general on Internet). Solutions of such challenges are crucial to support/improve smooth experience of users with the Web as well as their decision making. Web challenges are strongly associated with intelligent algorithms of various kind and purpose acting as clever solutions simplifying many tasks such that mining users’ data and relevant data sources with associated semantics, extracting users’ contexts and predicting their interests leading to improved data providers, user experience and semantics as proper knowledge representations.

Workshop Program Committee

Jaroslav Kuchař, ČVUT Praha
Peter Vojtáš, UK Praha
Ladislav Peška, UK Praha

Milan Dojchinovski, FIT ČVUT Praha
Tomáš Vitvar, FIT ČVUT Praha
Tomáš Horváth, ELTE Budapest
László Grad-Gyenge, Creo Group, Budapest
András Micsik, MTA SZTAKI, Budapest
Tomáš Kliegr, VŠE Praha

S. Krajči (ed.): ITAT 2018 Proceedings, pp. 146–151
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Štepán Balcar

Preference learning by matrix factorization on island models

Štěpán Balcar

Charles University, Faculty of Mathematics and Physics, Czech Republic, Prague
Stepan.Balcar@mff.cuni.cz

Abstract: This paper presents island models, methods, implementation and experiments connecting stochastic optimization methods and recommendation task (collaborative one). Our models and methods are based on matrix factorization. Parallel run of methods optimizes the RMSE metric from an island model point-of-view.

This paper comments on architecture and some implementation decisions.

We dealt with two research hypotheses. First, whether island models bring always improvement. We will show that almost always yes. Second, whether evolutionary algorithm does or does not always find the best solution. This will be confirmed only on smaller data. Experiments were provided on Movie Lens 100k and 1M data.

1 Introduction

This paper studies recommender systems, especially learning user/customer preferences. Main idea of this paper is to connect this study to evolutionary island models. Here, island models are a computational tool for matrix factorization.

Instances of one stochastic optimization method run in parallel on island models, searching the same state space. Such a method traverses the state space of solutions. The actual state they visit is influenced by (mutually independent) stochasticity. Hence, different instances can visit different parts of the state space. Island model parallelization is based on cooperation of methods during the computation.

This resembles ensemble and/or hybrid learning, where different optimization methods are:

- run in parallel
- allowed to run in different state spaces

The solution (usually one of them) is chosen at the end by an additional aggregation/decision method.

Using evolutionary computing for recommendation is surveyed in the paper of Horvath [1]. This survey shows that usage of evolutionary methods is quite frequent in recommendation systems (see also [2]). One of approaches is model based. Our approach uses matrix factorization and hence the model is constituted by factors. In [1] they mention only one paper [3] where evolution individuals are matrix factors. Authors of the article [3] provide results on ML100k data on minimizing squared error depending on size of population and probability of mutation (other parameters are fixed).

In this paper, parameters of our methods were chosen after experiments on sample data. We will try to improve RMSE using parallelization. We will provide results on both ML100k and ML1M data.

Main contribution of this paper is:

- Multiple island model brings statistically significant improvement of recommendation in comparison to single instance optimization.
- Our implementation is able to handle individuals (matrix factors). Size of our individuals is several orders bigger than usual in evolutionary computation.

This paper is organized as follows: Chapter 2 "Related work" concerns recommender systems and matrix factorization; evolutionary algorithms and island model. Chapter 3 "Methods, models and parameters" sketches our view of stochastic optimization methods; parameters and settings of island model used in tests. Next section "Data" describes realization of experiments and design of tests and computing resources. Finally section 4 "Results" gives both numeric and graphic presentation of our results and discusses confirmation of our hypotheses. After conclusions paper ends with Future work.

2 Related work

This section gives basic notation for recommender systems, evolutionary computation used and overviews some relevant literature.

2.1 Recommender systems and matrix factorization

It was the Netflix price (announced in October 2006) which excited public and changed the landscape of recommender systems area research.

The BellKor squad included Chris Volinsky and his AT&T colleagues Robert Bell and Yehuda Koren, along with four other engineers from the United States, Austria, Canada and Israel (BellKor's Pragmatic Chaos) on June 26, 2009, finally crossed the 10% finish line. Main ingredient of winning solution has been matrix factorization (MF), see e.g. [4].

We focus on latent model based recommendation - which is a part of collaborative filtering technique (Co),

asked for in Netflix Price competition. Co was introduced by [4].

Let us denote $\mathscr{U}$ set of user ids and $\mathscr{I}$ the set of item ids. Input data can be viewed as partial function $r : \mathscr{U} \times \mathscr{I} \longrightarrow \mathscr{P}$, here $\mathscr{P}$ is the preference scale (e.g. one to five stars or real numbers). Alternative representation is in the form of algebraic matrix $\mathbb{R} \in \mathscr{P}^{|\mathscr{U}|\times|\mathscr{I}|}$ with dimensions representing users and items respectively. Rating of user $i \in \mathscr{U}$ of an item $j \in \mathscr{I}$ is content of corresponding cell of matrix $r_{ij} \in \mathscr{P}$. Usually this matrix is very sparse, so in practice input data are represented in a form of a database table R with schema $R(uid = user\ id, iid = item\ id, rating)$. Although implementation uses database table formal model of [4] description is easier in the language of matrices. The matrix $\mathbb{R}$ is factorized to lower dimensional representation

$$\mathbb{R} \approx \mathbb{U} \times \mathbb{I}^{\intercal} = \hat{\mathbb{R}} \tag{1}$$

where $\mathbb{U} \in \mathscr{P}^{|\mathscr{U}|\times d}$, $\mathbb{I} \in \mathscr{P}^{|\mathscr{I}|\times d}$ are user and item latent factors matrices and $\times$ is a matrix product. d is much smaller than $|\mathscr{U}|$ and $|\mathscr{I}|$. Approximation of $\mathbb{R}$ by $\hat{\mathbb{R}}$ is measured by

$$e_{ij}^2 = (r_{ij} - \hat{r_{ij}})^2 = \left(r_{ij} - \sum_{k=1}^{d} u_{ik} i_{jk} \right) \tag{2}$$

here $\hat{r_{ij}}$ will serve as approximation of value r_{ij}.

Our main optimization method is SGD - stochastic gradient descent method. For other approaches and dimensions of research in recommender systems we reffer to [5].

2.2 Evolutionary algorithms

Evolutionary algorithms became popular after 1970, thanks to John Holland [6] as a means for solving optimization problems. The solution is represented as an individual, the set of individuals forms the population. Evolutionary algorithm is formed by phases: initialization, selection (parent selection for next generation offspring), crossover, mutation and evaluation of the whole population. The stochastic computing of individuals seeks convergence to the global optimum.

The contribution of evolutionary algorithms in the area of recommender systems was reviewed by Horvath de Carvalho in [1]. This review analyzed more than 65 research papers using EC techniques for user recommendations. It analyzed different approaches according to five aspects: (i) recommendation technique used, (ii) datasets employed in the experiments and (iii) evaluation methods used in the experiments, (iv) baselines used to compare with the proposed methods and (v) reproducibility of the research.

Most of nature-inspired algorithms reviewed in [1] find application in solving partial problems such as feature weighting extraction, model based approach, clustering and function learning.

Computing latent factor models by using of evolutionary algorithms has emerged as a possible approach [3]. Available publications do not mention the parallelization of evolutionary algorithms in combination with the computation of latent factors.

Motivated by this, in our paper individuals are represented by concatenation of $\mathbb{U}^{\intercal}$ and $\mathbb{I}^{\intercal}$ (like a worm d-thick and $|\mathscr{U}| + |\mathscr{I}|$ long)

$$\begin{bmatrix} u_{11} & \dots & u_{1|\mathscr{U}|} & i_{11} & \dots & i_{1|\mathscr{I}|} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ x_{d1} & \dots & u_{d|\mathscr{U}|} & i_{d1} & \dots & x_{d|\mathscr{I}|} \end{bmatrix}$$

Figure 1: Individual represented by latent vector of users and latent vector of items

In our experiments we use the RMSE as the fitness function

$$\sqrt{\frac{\sum_{i=1}^{|\mathscr{U}|} \sum_{j=1}^{|\mathscr{I}|} e_{ij}^2}{|\mathscr{U}| * |\mathscr{I}|}} \tag{3}$$

2.3 Island model

Island models are one of approaches how parallelize optimization algorithms. Their characterization is that there is no central master and populations are distributed. Island model consists of parallel algorithms enriched by sending and accepting migrants. Migrants are individuals from other islands population. The hope is that this propagation of genetic material can speed up convergence and escape from local optima trap.

Parameters of island model are frequency of sending-receiving migrants and the way how the migrant is chosen from local population (remember that parameters of methods are fixed and are chosen after experiments on a sample data). Optimal choice of these parameters depends on the type of optimization method and is subject of study. These aspects of island models are studied in [7].

Specific type of island models, heterogeneous, were used in the application of multi-criteria optimization by Martin Pilát and Roman Neruda [8]. They designed a new algorithm of MOGASOLS (Multiobjective Genetic Algorithm with Single Objective Local Search) combining multi-criteria genetic algorithm (MOGA) with a single-criteria genetic algorithm (SOGA). It is proved that parallelization of time-consuming method MOGA achieves worse results than parallel running MOGA and SOGA methods. They were tested in NSGA-II [9] and IBEA [10] algorithms. This was further developed in [11].

The first who come up with heterogeneous island models (HeIM), the way we understand them, was Martin Pilat ([11]). In these models, the islands may carry diverse computational methods, differing not only in parameters but also in the structure of the whole stochastic algorithm [11]. For the choice of optimal methods for the given problem is

responsible the central planner, which replaces unsuccessful method by more successful methods during the whole computation. In publication [11] the original Balcar's tool [12] was used.

In this paper this tool is used with homogeneous islands to find optimal user and item latent factor.

Then different optimization techniques are used to converge with factor to optimal one. For illustration we give formulas for stochastic gradient descent. Recall e_{ij}^2, then next generation values of user and item factors equal to

$$u_{ik}^{'} = u_{ik} + \alpha \frac{\partial}{\partial u_{ik}} e_{ij}^2 \qquad i_{kj}^{'} = i_{kj} + \alpha \frac{\partial}{\partial i_{kj}} e_{ij}^2 \qquad (4)$$

Note, that in our implementation we do not consider normalization factor, this is left for future. Nevertheless, some normalization effect is obtained by migration and synergy of islands. α is not fixed, just bounded from above. More on our system is in [12].

3 Methods, models and parameters

Island models where originally developed for parallelization of evolutionary algorithms. In this paper we will use also other stochastic optimization methods.

3.1 Stochastic optimization methods

Evolutionary algorithms try to balance trade off between exploration and exploitation. This property should help evolutionary algorithm to escape from local extremes and simultaneously converge to optimal solution. For this ability they pay a higher time complexity because of parallel development of bigger population. Because of this fact, on some types of problems, the winners are other stochastic optimization methods. An example is TSP solution by hill climbing with 2-opt ([13]). So, this is the main reason we consider several additional stochastic optimization methods (see Table 1).

All these methods use the stochastic gradient descent. A general link to description of stochastic optimization methods is [14].

Our implementation of stochastic optimization methods was changed in two ways. First, we had to create operators to enable them to work with latent factor based individuals. Second, methods were modified for parallel run in island models with migration. They were enriched with ability to cooperate. They receive individuals and enrich their local population. Please notice, that only evolution and differential evolution have bigger population. Remaining methods have only one individual population. In this case enrichment means replacement. They provide their solution from local population to neighboring islands. Methods manipulate incoming individuals in concordance with basic algorithm idea. E.g. the tabu search method inserts a newcomer to the tabu set.

Individual equals solution. Solution is represented by pair of latent factors of users and items (Figure 1). Multiplying of these latent factor we get a matrix which represents our estimation of ratings. Length of first vector is the number of users and the length of second vector equals number of items. Size of the individual depends on size of input problem. Width of latent vector is an optional parameter.

3.2 Parameters and settings of island model used in tests

Now we describe island model - the environment in which the above methods will be parallelized. It is the environment which ensures dissemination of genetic material. The main tool for this is the migration. This migration does not mean only exchange of best solutions. We would like to spread across the system (between islands) genetic material which has the potential to contribute to further improvement of population quality and to speed-up the convergence. Migration is inspired by processes in nature ([15], [16]). Most of island models exchange migrants in a synchronized way. In our system exchange of migrants is not synchronized - so in fact our islands are more general.

Key parameter of island models , as nature shows ([17]), is the frequency of migration and size of local populations. The bigger the frequency of migration the bigger is the chance that islands will help each other. On other side each hardware and software is limited by data through put.

Matrix factorization needs migration of much bigger individuals than e.g. continuous optimization, where individual is a point in an n-dimensional space. In this paper we had to change architecture of model used in [11]. In [11] input problems were TSP (traveling salesman problem of size cca. 1000 cities), BP (bin packing, one dimensional with 1000 items), CO (continuous optimization, 10-dimensional function) and vertex cover (1000 vertexes). Solution of preference learning by matrix factorization on island models is represented of much bigger individuals, rough estimation of our state space is $> \#\text{TSP}^{20}$. Evolutionary algorithms use incoming individuals in groups and after a while. Hence, it is advantageous to store them in a front. As far as individuals are big and memory is limited we had to limit the size of fronts (see Table 2).

3.3 Data

We used data from the movie recommendation domain in the experiments. The effectiveness of parallelization has been verified on datasets ml-100k[1] and ml-1m[2]. Datasets are formed by movie evaluation (1-5 stars), for trials we use the trinity (user, movie, rating).

The training set consists of four-fifths from the dataset, the rest is the test set. Counting derivatives for SGD lever-

[1] http://grouplens.org/datasets/movielens/100k/

[2] https://grouplens.org/datasets/movielens/1m/

Algorithms	Tool	Parameters
HillClimbing	SGD random rating from each line	numberOfNeighbors=10
RandomSearch	generation of latent vectors	
Evolution	uniform crossover + SGD	popSize=10, mutationRate=0.9, crossRate=0.1, parentSelector=CompareTwoRandomSelectors
BruteForce	SGD according to the I-th rating	
TabuSearch	SGD random rating from each line	tabuModelSize=50
Sim.Annealing	SGD random rating from each line	temperature=10000, coolingRate=0.002
Diff.Evolution	differential crossover	popSize=50, F=0.25

Table 1: Methods and parameters

Parameter	value
Number of iterations	50, period = 60 seconds
Number of islands	4 (AMD Opteron 6376)
Neighbors of method	3 (distributed to everyone)
Migration frequency	5 seconds

Table 2: Parameters of the island model

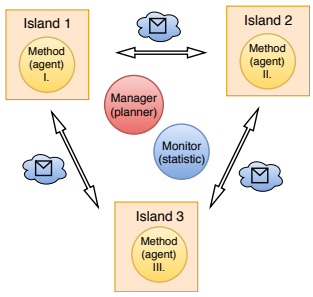


Figure 2: Architecture of system

ages the training set. The test set is used to evaluate individuals.

3.4 Realization of the experiment

Our main idea is to increase stochasticity of searching the state space (which is enormous for movie data). First level of stochasticity is enabled by stochastic optimization method. Second level is enabled by several independent islands. The third level is attained by migration. Our system enables all of these. Moreover, all of these run in parallel with mutually assisting methods (not competitive).

In order to be able to obtain the best solution from such a computation resource, the computation must be continuously monitored (Figure 2). Solutions represent individuals that are unpredictably moving across the island model. We can never know when and where the best solution will appear (sometimes the best solution does not appear at the end, see Figure 3).

Our implementation uses agent middle-ware Jade[3] for achievement of higher adaptivity of methods (in our three levels of stochasticity).

Here we were facing main decision. Whether to prefer higher adaptivity (based e.g. on Jade) or better use of effectiveness of specific hardware (based e.g. on C). From pure experimental curiosity we went along the agent based middle-ware framework direction.

The central brain of the multi-agent based system is a manager of migration which directs the computation and measures genetic material during evolution.

[3] http://jade.tilab.com/

The use of this system is the implementation of methods as an agent, who can develop methods as adaptive computing resources. The infrastructure of the system is made up of two central points, by Agent-manager that manages computation and by Agent-monitor, that monitors genetic material in the system.

Software allows multiple ways of monitoring. The monitor statistically processes the quality of the individuals. Another way of observing what happens in the system is to produce the pedigrees of an individual. The basic idea is to enrich every individual of the pedigree that contains information on the involvement of concrete islands in its creation.

For our experiments, were used statistics that are computed from each monitored individuals in one iteration of planner. We will present the results as a measure of the system, which is monitoring follow-planner-iterations.

Our evolutionary methods (Table 1) use uniform crossover. In phase of mutation they apply stochastic gradient descent on a randomly chosen rating.

Differential evolution combines 3 latent vectors (individuals which are models/solutions) $\mathbb{LV}_1$ which is a concatenation of $\mathbb{U}_1^\intercal$ and $\mathbb{I}_1^\intercal$, similarly $\mathbb{LV}_2$ and $\mathbb{LV}_3$. Result of the differential operator is latent vector

$$\mathbb{LV}_{new} = \mathbb{LV}_1 + F * (\mathbb{LV}_2 - \mathbb{LV}_3)$$

here F is 0.25.

3.5 Test design and computing resources

Two pairs of tests were created, comparing single methods and island models, differing in the size of the input Movieland dataset. The width of the latent vector was set to 10 (best after initial experiments on smaller samples). The tests run on all initial parameter settings (Table 3) 9 times. As far as our computations are nondeterministic this makes difference (see Figures 4 and 5). These are computationally demanding calculations.

Parameter	value
Number of runs	9
Computing nodes	AMD Opteron 6376, 64GB memory
Latent factor width	10
Datasets	ml-100k, ml-1m

Table 3: Parameters of the test

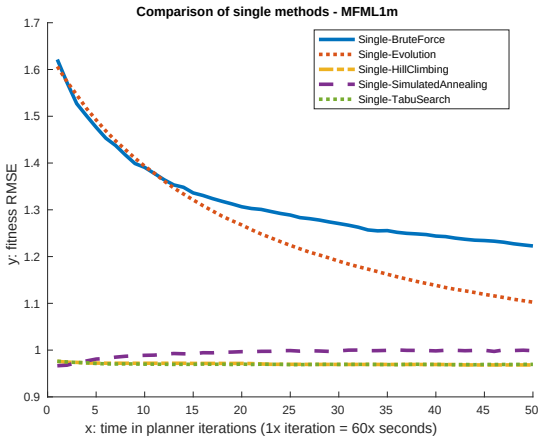


Figure 3: Methods ML1m investigation of median run

4 Results

Our experiments validated two hypotheses. First is that evolution does not necessary give best solution and second that island models improve results. We will present results for two datasets separately. We will show the best solution of 9 runs and average of all runs (for distribution see Figures 4 and 5).

On the smaller data set the winner is always Evolution and Islands give always better solution.

On the bigger dataset the single island winner is simulated annealing (in both minimum and average). In Islands the winner is hill climbing (in both minimum and average). On bigger data we can not always observe advantage brought by parallelization by islands and migration. This can be observed especially by simulated annealing. One possible explanation is that hill climbing does not risk that much going in wrong direction as simulated annealing is doing (sometimes). Bigger data bring bigger state space and hence risk is necessary, but 50 iterations of the planner is probably not enough. In future research we will run island models with more iterations.

5 Conclusions

We proved that island models are a good computing tool for recommender systems. Our experiments have shown following. Island models brought clear improvement on smaller data set. On bigger data, it is also true except of simulated annealing. Moreover on bigger data evolution does not find best solution.

Our implementation is publicly available on Github[4] and hence enables repeatability of our experiments (see [1], requirement (v)).

[4] https://github.com/sbalcar/distributedea

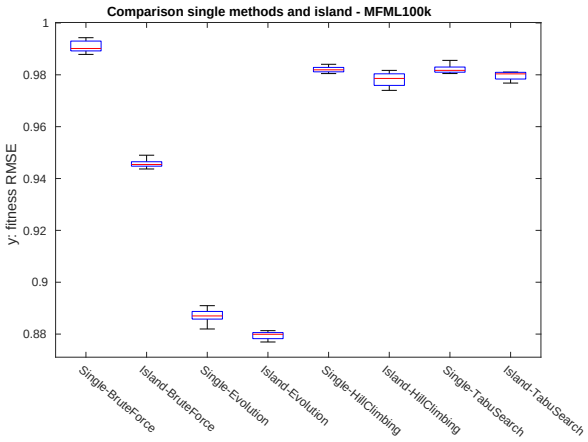


Figure 4: Methods ML100k boxplot of results

6 Future work

In our future work we plan to extend this to heterogeneous island models. We also plan to develop new planners which will be specifically designed for recommendation domain. We would like to consider also islands with statistical learning methods. Challenge is extension to content based recommendation.

References

[1] Tomáš Horváth and André C. P. L. F. de Carvalho. Evolutionary computing in recommender systems: a review of recent research. *Natural Computing*, 16(3):441–462, 2017.

[2] M. Rezaei and R. Boostani. Using genetic algorithm to enhance nonnegative matrix factorization initialization. In *2010 6th Iranian Conference on Machine Vision and Image Processing*, pages 1–5, Oct 2010.

Methods	Single-min	Islands-min	Single-average	Islands-average
BruteForce	0.98787115	0.94367838	0.99088728	0.94571058
DifferentialEvolution	1.50105714	1.49096267	1.51771324	1.49957429
Evolution	**0.88196787**	**0.87695296**	**0.88705747**	**0.87952888**
HillClimbing	0.98047412	0.97400051	0.98207866	0.97824895
RandomSearch	1.60448414	1.58333447	1.61703977	1.60802325
SimulatedAnnealing	1.06110779	1.03108626	1.07797515	1.03806128
TabuSearch	0.98048607	0.97681064	0.98217126	0.97963446

Table 4: Comparison of single methods and island models: Dataset - MFML100k, Runs - 9

Methods	Single-min	Islands-min	Single-average	Islands-average
BruteForce	1.22268511	1.18410586	1.23279212	1.2033589
DifferentialEvolution	1.55227103	1.55057106	1.55780106	1.55422412
Evolution	1.10254503	1.07069968	1.13526207	1.09140823
HillClimbing	0.96832015	**0.96727131**	0.97065502	**0.96840234**
RandomSearch	1.65483811	1.66265835	1.66738744	1.6660426
SimulatedAnnealing	**0.96655732**	0.97118289	**0.97048761**	0.9729405
TabuSearch	0.96873215	0.96735123	0.97106628	0.96854622

Table 5: Comparison of single methods and island models: Dataset - MFML1m, Runs - 9 (in red there are violations of island improvement)

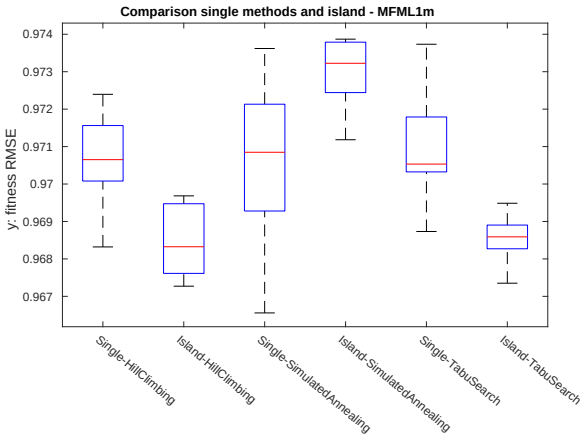


Figure 5: Methods ML1m boxplot of results

[3] Dariush Zandi Navgaran, Parham Moradi, and Fardin Akhlaghian. Evolutionary based matrix factorization method for collaborative filtering systems. *2013 21st ICEE*, pages 1–5, 2013.

[4] Y Koren, R Bell, and Volinsky C. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[5] L. Peska and P. Vojtas. Using implicit preference relations to improve recommender systems. *Journal on Data Semantics*, 6,1:15–30, 2017.

[6] John H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, 1992.

[7] Darrell Whitley, Soraya Rana, and Robert B. Heckendorn. The island model genetic algorithm: On separability, population size and convergence. *J.Comp.Inf.Tech.*, 7:33–47, 1998.

[8] Martin Pilát and Roman Neruda. Combining multiobjective and single-objective genetic algorithms in heterogeneous island model. In *CEC 2010*, pages 1–8, 2010.

[9] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Trans. Evol. Comp*, 6(2):182–197, 2002.

[10] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. Spea2: Improving the strength pareto evolutionary algorithm. Technical report, ETH Zurich, 2001.

[11] S Balcar and M Pilat. Heterogeneous island model with replanning of methods. In *GECCO'18, accepted as poster*, 2018.

[12] Stepan Balcar. Heterogeneous island model with replanning of methods (in czech). *Master thesis, Martin Pilát supervisor*, 2017.

[13] G. A. Croes. A method for solving traveling salesman problems. *Operations Research*, page 791–812, 1958.

[14] R.K. Arora. *Optimization: Algorithms and Applications*. CRC Press, 2015.

[15] Jinliang Wang and Michael C. Whitlock. Estimating effective population size and migration rates from genetic samples over space and time. *Genetics*, 163(1):429–446, 2003.

[16] William Astle and David J. Balding. Population structure and cryptic relatedness in genetic association studies. *Statist. Sci.*, 24(4):451–471, 11 2009.

[17] John Wakeley. The coalescent in an island model of population subdivision with variation among demes. *Theoretical Population Biology*, 59(2):133 – 144, 2001.

Acknowledgements. This work was supported by a Ph.D grant SVV2018-260451 under supervision of Peter Vojtas.

S. Krajči (ed.): ITAT 2018 Proceedings, pp. 152–156
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Peter Gurský, Matej Perejda, and Dávid Varga

Semi-automatic annotation of e-shops

Peter Gurský, Matej Perejda ,Dávid Varga
Institute of Computer Science
Faculty of Science, P.J.Šafárik University in Košice
Jesenná 5, 040 01 Košice, Slovakia
peter.gursky@upjs.sk, {matej.perejda, david.varga}@student.upjs.sk

Abstract. *Extraction from web pages becomes a very popular way to acquire important data for better decision process. Acquisition of structured data from a new web portal requires an annotation of web pages of the portal to allocate the location and the type of the information. We present methods for semi-automatic annotation of e-shops' content, to create rules for extraction. The methods are implemented in Chrome extension named Exago. The aim of these methods is to generate XPaths and regular expressions. We use positive and negative examples to further specify which of the generated XPaths should be used for extraction. The annotation methods are tested on real data and the results show a high success rate.*

1 Introduction

Web scraping is a complex process that consists of web pages annotation, crawling, data extraction and data processing. This kind of data acquisition is very valuable for decision process, because it can provide data from various sources and process them together. Project Kapsa [1] deals with extraction and unification of information from web pages, focusing on products on e-shops. The aim of the project is a creation and management of a collection of products which are offered by e-shops. In this paper, we focus on the first step of the web scraping process, the web pages annotation.

The annotation has two main goals: recognition of relevant page on portal and identification of positions of relevant pages, where the data of our interest are.

The positions of relevant data are usually specified by XPaths of HTML source or by regular expressions, which are used by extractor on each relevant page. It is also possible to extract data using a procedural script. Writing complex XPaths or regular expressions, as well as a creation of scripts is not an easy task. It requires the annotator to be an IT expert.

Our goal is to make the annotation process of e-shops easier and possible for ordinary person. We would like to offer in our Chrome extension Exago[1] satisfactory results of web scraping with annotation made only by mouse clicking.

2 State of the art

Web annotation and extraction systems can be categorized to four groups [2]. Manual systems [3, 4] require programming in some (pseudo) language. Automatically constructed extractors [5, 6] create extraction system based on complete user annotation and examples of extracted data from several pages. Automatically constructed extractors with partial user support [7, 8] create extraction system without the need of extraction examples. Automatic extractors with no user support [9, 10] analyze repeating patterns on web pages, and extract every data that seem to be interesting. Our tool Exago can be included in automatically constructed extractors with partial user support.

Currently there are more than 50 web scrapers available on the internet. We have tried to use all of them to product data extraction from two e-shops Alza.sk and Heureka.sk. The majority of them were not able to extract the product data. The web scrapers that are at least partially applicable to product data extraction are [11-28], so we examined these web scrapers more deeply. Majority of these tools provide the generation of one XPath by clicking on element on the page. Some tools hide this functionality and do not show the final XPath [12, 18, 19, 21, 22, 24, 25, 26], the others [11, 13, 15, 17, 20, 23, 27, 28] allow the modification of the XPath manually. The rest of them [14, 16] provides manual insertion of XPath only. None of the tools provide regular expression generation, but some of them allow writing regular expressions manually.

3 Methods for annotation

Having an HTML element containing the relevant data we can easily create an XPath as a path from the root element. The XPath points to the target element and it is used during the extraction process on this page. However on the similar page (e.g. the page about different product on e-shop), the created XPath can be unsuccessful – either it finds no element or an element with different kind of data, because the HTML source tree can be slightly different (e.g. missing subtree, different highlighting of texts etc.) than the one of the annotated product.

Fortunately, many XPaths can be created that point to the same element. The tree navigation of XPaths can be based on elements' attributes, order of element between its siblings, various conditions and more. Choosing the right navigation of XPath increases the success rate of extraction considerably. The problem is that an unexperienced user is not able to choose the right XPath from the hundreds of possibilities. Therefore we provide semi-automatic XPath and regular expression generation and interactive selection of the right rules to make the annotation process easier.

3.1 XPath generation

XPath is a query language that is used to select nodes of XML DOM model. All browsers create a DOM model out of HTML file as the first step of page processing. XPath language provides various approaches to specify a starting node(s) and traversal of the DOM model. Creator of XPath expression can utilize tag names, tag attributes and their values, order of the elements, navigation functions, and conditions with build-in functions. Such variability allows many XPaths to localize the same node.

Annotator's goal is to specify the position of relevant data that can be universal for all pages of the same type (created from the same HTML template). In our case, we

focus primarily on pages, where the details of the products are located. Unfortunately, when the template is combined with the structured data of products, to create final detail pages, the differences between result pages eventuate in variety of HTML tree structures. They can vary in element attributes as well as in absence of whole subtrees. Such differences cause many XPaths, which work on one page, fail on other page. They can point to different or no nodes, while the corresponding data is still present somewhere on the page.

It is impossible to know, which parts of the template are on all result pages, without complex analysis of pages, because the templates are not public. Therefore we don't know which elements or attributes can be used as navigation points of universal XPath. Annotation experts usually examine various HTMLs of result pages and create the universal XPath manually.

```
<h1>Samsung 24FDX</h1>
<h2>Specification</h2>
<table id="product parameters">
  <tbody>
    <tr>
      <td style="color:powderblue;">Producer</td>
      <td style="color:pink;">Samsung</td>
    </tr>
    <tr>
      <td style="color:powderblue;">Model</td>
      <td style="color:pink;">24FDX</td>
    </tr>
  </tbody>
</table>
```

Fig. 1. Part of HTML source example

Our approach generates several possible XPaths as possible candidates to final universal XPath. Consider that annotator wants to create an XPath leading to element with value "24FDX" in HTML source on Figure 1. In this example, the result of the method used for generating XPaths is a set of 36 different correct XPaths, which point to the same element. The shortest ones of them are listed below:

- //tr[2]/td[2]
- //tr[last()]/td[last()]
- //tr[last()]/td[2]
- //tr[2]/td[last()]
- //tr[2]/*[@style="color:pink;"]
- //tr[last()]/*[@style="color:pink;"]

XPaths are generated gradually along the path from the clicked element to a specified root element occurs. If there is no root element specified, the root element of HTML document is chosen to be the element where generation stops. At every current element during the generation, all attributes, the name and the order between siblings of this element are combined to create different XPaths.

The extractor process, which extracts the data from all similar pages, needs only one XPath per each value position. Our approach in Exago tool [1] chooses the shortest XPath as the result candidate by default. The annotation process is a process where annotator determines whether the candidate XPath is the most universal one, to be chosen for extraction.

During the annotation process, the annotator can navigate to other similar page (e.g. the page about other product) and check out the success of the chosen XPath. There are three possibilities:

1. The element is correctly found using the XPath and no changes need to be done.
2. The XPath addresses no element and annotator can mark the correct element as **the positive example**.
3. The XPath addresses different element and annotator can mark the addressed element as **the negative example**.

When the annotator marks the positive or negative example, by clicking on an HTML element, the method for generating XPaths generates new set of XPaths to this element. Let this set of XPaths be named as B, and the original set as A. In case of the positive example, the new set of XPaths, that work fine on both pages is A∩B. In case of negative example, the new set is A-B.

After some iterations of this procedure, the result set contains only XPaths that work on all pages. The annotator can choose any of them to be part of final extraction rule, or just keep the default one. As it was mentioned before, our Exago tool chooses the shortest XPath by default.

3.2 Regular expression generation

XPath is a very capable language at locating the whole element of HTML source. There are cases, when we want to extract a value only from a part of the element, or a value spreads across two or more elements. In this case, XPath is unusable.

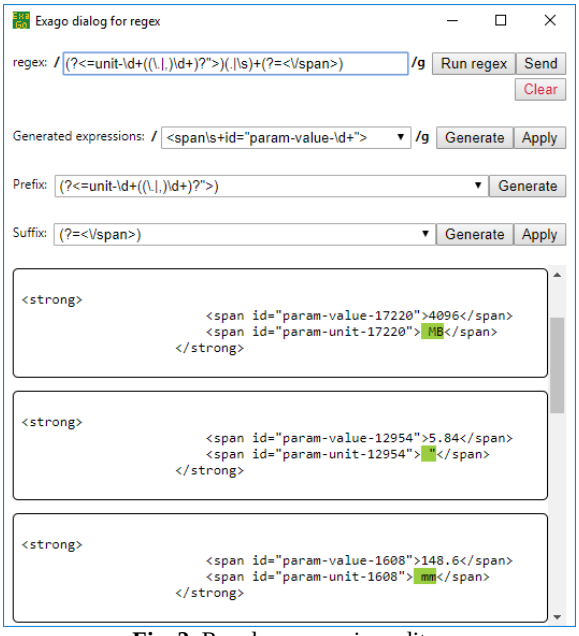


Fig. 2. Regular expression editor

In Exago, we combine XPaths and regular expressions, if needed. XPath localizes the HTML part in which the regular expression can be used. It is also possible, that the XPath localizes more than one element, and regular

Table 1. Annotation accuracy analysis

	first genarated XPath			our approach		
URL of e-shop	Successful annotation of elements	Success rate	Use of regular expressions	Successful annotation of elements	Success rate	Use of regular expressions
gymbeam.sk	10 out of 14	71%	No	11 out of 14	79%	No
vivantis.sk	8 out of 12	67%	No	10 out of 12	83%	Yes
bestbuy.com	5 out of 14	36%	No	12 out of 14	86%	Yes
martinus.sk	8 out of 14	57%	No	12 out of 14	86%	Yes
nay.sk	10 out of 14	71%	No	12 out of 14	86%	Yes
insportline.cz	9 out of 14	64%	No	12 out of 14	89%	Yes
target.com	6 out of 9	67%	No	8 out of 9	89%	Yes
eshop.eta.cz	8 out of 11	73%	No	10 out of 11	91%	Yes
mall.sk	9 out of 11	82%	No	10 out of 11	91%	Yes
notino.sk	4 out of 11	36%	No	10 out of 11	91%	Yes
pantarhei.sk	8 out of 11	73%	No	10 out of 11	91%	Yes
obi.cz	3 out of 14	21%	No	13 out of 14	93%	Yes
andreashop.sk	9 out of 11	82%	No	11 out of 11	100%	No
decathlon.sk	12 out of 13	92%	No	13 out of 13	100%	No
hej.sk	9 out of 11	82%	No	11 out of 11	100%	No
heureka.sk	5 out of 12	42%	No	12 out of 12	100%	Yes
hornbach.sk	5 out of 10	50%	No	10 out of 10	100%	Yes
radioshack.com	9 out of 12	75%	No	12 out of 12	100%	No
rajdazdnikov.sk	0 out of 11	0%	No	11 out of 11	100%	Yes
zoohit.sk	5 out of 11	45%	No	11 out of 11	100%	Yes
Total	**142** out of **240**	**59%**	-	**221** out of **240**	**93%**	-

expression can be used in all of them to find out the target value.

Regular expression is an effective tool for extraction of a substring based on complex conditions with quite complicated syntax. Sometimes, even IT experts have a hard time at constructing more complex regular expressions.

In Exago, we created regular expression editor (Fig. 2) that generates multiple regular expressions using mouse events. . The process of generating regular expressions can be performed only by clicking on buttons in the editor and highlighting the text needed for extraction. Therefore, the editor allows a less experienced user to create regular expressions, and see the result. On the other hand, there is still a possibility to edit generated regular expressions or write custom ones.

Our editor supports two approaches. First, user can select the target text and click on the first Generate button. Exago generates several regular expressions, collected in the combo box. The method that generates the expressions tries to generalize the two most common text parts:

- spaces and other whitespace characters are converted to expression *\s* or *\s+*,
- numbers are converted to *\d+* or *\d+((\.|,)\d+)?*, which covers also decimal numbers.

The second approach to regular expression generation is selection of prefix and suffix of the target value and hitting the appropriate "Generate" button. In Figure 2, user selected text "*unit-17220>*" as the prefix and "**" as the suffix and generated the related regular expressions. Using the chosen regular expressions, a combined expression is created and written in the text field on the top of the screen. The result of the regular expression search is emphasized on the bottom with green background.

4 Experiments

In this section we analyze the accuracy of annotation of our new version of Exago. The new version is using the semi-automatic annotation based on positive and negative examples and generating regular expressions. The tests compare our approach with the usual approach of element localization used in other web scrapers and our previous version of Exago, i.e. generation of one XPath per value.

This analysis has been done on 20 different e-shops. During the testing phase, every annotation has been realized only by clicking with mouse on HTML elements and Exago components. No manual editing of XPaths and regular expressions has been used, in order to compare the old and the new approach. The table on Table 1 shows the results of each annotation per e-shop ordered by success rate.

We measure following aspects for both approaches:

- successful annotation of elements – the number of successfully annotated elements or values compared

to the number of all elements or values that could be annotated,
- success rate – percentage of correctly addressed elements and values among annotated elements and values
- use of regular expressions – information about the use of generated regular expressions during annotation; with every use of regular expressions in Exago, regular expressions have been used in combination with XPaths.

With the new approach, we have achieved success rate of 100 % in 8 internet shops. In remaining 12 e-shops we have not been able to achieve the 100 % success rate because of the following reasons:

- in 5 cases, the HTML structures of detail web pages in each e-shop have varied too much,
- in 5 cases, lists of product parameters have been divided into more elements unrelated to each other,
- in 3 cases, complications occurred during annotation of images that used dynamic styles of their presentation on a web page,
- in 3 cases, we have not been able to annotate prices of products, because e-shops displayed sale prices in different elements compared to non-sale prices, whilst both of these prices have been present on a web page,
- in 3 cases, we have not accomplished to annotate product ratings represented by pictures without any further information given, for example, amount of stars awarded.

Success rate average of annotation of information about products has been 59% in the case of common approach. With the new version of Exago we have achieved the average success rate of 93%. This growth has been achieved with the help of generating regular expressions and the functionality of positive and negative examples.

Some of the unsuccessful cases could be eliminated by manual editing of XPaths and regular expressions. For example, in the cases when internet shops presented more lists of product parameters, we would be able to manually create a regular expression that would address all types of elements representing these lists.

5 Conclusions

This paper deals with improvement of the commonly used data localization process in web annotation by implementing the semi-automatic annotation. We have created a plug-in module Chrome Extension named Exago containing this functionality. With the help of methods for generating XPaths, generating regular expressions and the functionality of positive and negative examples, we are able to annotate more information on detail web pages of products, compared to the previous version of Exago.

During the annotation accuracy analysis, we have been comparing the common approach with the new one, and discovered that the success rate has grown from 59% to 93%. We consider this result as notable, but there is still a room for improvement. One improvement could be creating a new component, which would be able to distinguish different types of detail web pages in one e-shop and use appropriate XPaths and regular expressions for annotation.

Another improvement could be more intelligent generation of XPaths by which we would generate XPaths faster and reduce the generation of very similar XPaths.

Designing methods for annotation of product ratings represented by pictures of stars or other objects would be also very benefiting. The solution to this problem could be counting matches of the regular expression addressing one star in the element containing these stars.

Adding a sale price as a new type of known value component would enable us to annotate sale prices as well as non-sale prices and grow our success rate per e-shop.

The new version of Exago could also be improved by solving a problem with image annotation, where images are being displayed in different dynamic styles. The solution could be some kind of a mechanism that would get all pictures from a detail web page, show them to the user, and the user would pick the image he wants. XPath addressing this image would be generated and used automatically.

References

[1] Project Kapsa, web page: http://kapsa.sk/
[2] B. Liu: Web Data Mining: Exploring Hyperlinks, contents and Using Data, Second edition, Springer 2011. ISBN 978-3-642-19459-7
[3] V. Crescenzi, G. Mecca: Grammars have exceptions. Information Systems, 23(8): 539-565, 1998.
[4] T. Furche, G. Gottlob, G. Grasso, C. Schallhart, A. Sellers: OXPath: A language for sca-lable data extraction, automation, and crawling on the deep web. The VLDB Journal 22(1): 47-72, 2013
[5] C. Hsu, M. Dung: Generating finite-state transducers for semi-structured data extraction from the Web. Information Systems, 1998, 23(8): p. 521-538.
[6] Muslea, S. Minton, C. Knoblock: A hierarchical approach to wrapper induction. In Proce-edings of Intl. Conf. on Autonomous Agents (AGENTS-1999) 1999.
[7] C.-H. Chang, S.-C. Kuo: OLERA: A semi-supervised approach for Web data extraction with visual support. IEEE Intelligent Systems, 19(6):56-64, 2004.
[8] Hogue, D. Karger: Thresher: Automating the Unwrapping of Semantic Content from the World Wide. Proceedings of the 14th International Conference on World Wide Web (WWW), Ja- pan, pp. 86-95, 2005.
[9] V. Crescenzi, G. Mecca, P. Merialdo: RoadRunner: towards automatic data extraction from large Web sites. Proceedings of the 26th International Conference on Very Large Database Systems (VLDB), Rome, Italy, pp. 109-118, 2001.
[10] Arasu, H. Garcia-Molina: Extracting structured data from Web pages. Proceedings of the ACM SIGMOD International Conference on Management of Data, San Diego, California, pp. 337- 348, 2003.
[11] Content Grabber. Web scraper available on-line: (https://contentgrabber.com/)
[12] Data Miner. Web scraper available on-line: (https://data-miner.io/)
[13] Helium Scraper. Web scraper available on-line: (http://www.heliumscraper.com/)
[14] Import.io. Web scraper available on-line: (https://www.import.io/)
[15] Mozenda. Web scraper available on-line: (https://www.mozenda.com/)
[16] ParseHub. Web scraper available on-line: (https://www.parsehub.com/)

[17] Scrapinghub (Portia) . Web scraper available on-line: (https://scrapinghub.com/)
[18] Web Scraper. Web scraper available on-line: (http://webscraper.io/)
[19] Agenty. Web scraper available on-line: (https://www.agenty.com)
[20] Data Toolbar. Web scraper available on-line: (http://datatoolbar.com/)
[21] Dexi.io. Web scraper available on-line: (https://dexi.io/)
[22] Easy Web Extract. Web scraper available on-line: (http://webextract.net/)
[23] Fminer. Web scraper available on-line: (http://www.fminer.com/)
[24] GetData.IO. Web scraper available on-line: (https://getdata.io/)
[25] Grepsr. Web scraper available on-line: (https://www.grepsr.com/chrome-extension/)
[26] Instant Data Scraper. Web scraper available on-line: (https://webrobots.io/instantdata/)
[27] Visual Web Ripper. Web scraper available on-line: (http://visualwebripper.com/)
[28] Web Sundew. Web scraper available on-line: (http://www.websundew.com)

S. Krajči (ed.): ITAT 2018 Proceedings, pp. 157–160
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Ladislav Peška and Hana Trojanová

Personalized Recommendations in Police Photo Lineup Assembling Task

Ladislav Peska
Department of Software Engineering
Faculty of Mathematics and Physics, Charles University, Prague
Czech Republic
peska@ksi.mff.cuni.cz

Hana Trojanova
Department of Psychology
Faculty of Arts, Charles University, Prague
Czech Republic
trojhanka@gmail.com

Abstract. *In this paper, we aim to present a novel application domain for recommender systems: police photo lineups. Photo lineups play a significant role in the eyewitness identification prosecution and subsequent conviction of suspects. Unfortunately, there are many cases where lineups have led to the conviction of an innocent persons. One of the key factors contributing to the incorrect identification is unfairly assembled (biased) lineups, i.e. that the suspect differs significantly from all other candidates. Although the process of assembling fair lineup is both highly important and time-consuming, only a handful of tools are available to simplify the task.*

We describe our work towards using recommender systems for the photo lineup assembling task. Initially, two non-personalized recommending methods were evaluated: one based on the visual descriptors of persons and the other their content-based attributes. Next, some personalized hybrid techniques combining both methods based on the feedback from forensic technicians were evaluated. Some of the personalized techniques significantly improved the results of both non-personalized techniques w.r.t. nDCG and recall@top-k.

1 Introduction

Evidence from eyewitnesses often plays a significant role in criminal proceedings. A very important part is the lineup, i.e., eyewitness identification of the perpetrator. Lineups may lead to the prosecution and subsequent conviction of the perpetrator. Yet there are cases where lineups can played a role in the conviction of an innocent suspect. This forensic method consists of the recognition of persons or things and thus is linked with a wide range of psychological processes such as perception, memory, and decision making. Those processes can be influenced by the lineup itself. In order to prevent witnesses from making incorrect identifications, the lineup assembling task is among the top research topics of the psychology of eyewitness identification [1, 4, 6, 9, 10].

The sources of error in eyewitness identifications are numerous. Some variables affecting error probability are on the side of the witness (e.g., level of attention, age or ethnicity) and the event (e.g., distance, lighting, time of the day) and in general cannot be controlled [6, 9]. Controllable variables include the method of questioning, identification procedure, interaction with investigators, and similar [9, 10].

One of the principal recommendations for inhibiting errors in identification is to assemble lineups according to the lineup fairness principle [1, 5]. Lineup fairness is usually assessed on the basis of data obtained from "mock witnesses" - people who have not seen the offender, but received a short description of him/her. Lineup fairness measures a bias against the suspect and defining the assembled lineup as fair if mock witnesses are unable to identify a suspect based only on a brief textual description. See Figure 1 for an example of a highly biased lineup.

Assembling photo lineups, i.e., finding candidates for filling the lineup for a particular suspect, according to the lineup fairness principle is a challenging and time-consuming task involving the exploration of large datasets of candidates. In the recent years, some research projects [4, 11] as well as commerce activities, e.g., *elineup.org*, aimed to simplify the process of eyewitness identifications. However, they mostly focused on the lineup administration and do not support intelligent lineup assembling.

From the point of view of recommender systems, lineup assembling is quite specific task for several reasons. Users of the system are respected experts, who assemble lineups regularly, although, usually, not on a daily bases. Therefore, we can expect a steady flow of feedback from long-term users. Also, each lineup assembling task is highly unique, i.e., the same suspect hardly ever appears in multiple lineups. Thus, some popular approaches incorporating collaborative filtering [2] or "*the wisdom of the crowd*" cannot be applied in this scenario. Last, but not least, the relevance judgement is highly based on the visual appearance and/or similarity of the suspect and lineup candidate.

In this paper, we describe our work in progress towards designing recommender systems aiding user to assemble fair lineups. In our previous work, we evaluated two non-personalized, item-based recommending strategies [8]. Based on the initial evaluation of non-personalized methods, we propose a content-based personalized approach combining both non-personalized techniques, aiming to re-

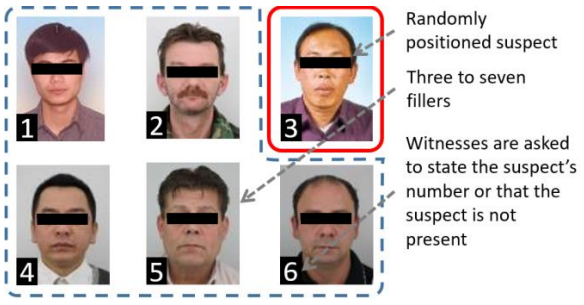


Figure 1: Example of an extremely biased lineup. Lineup usually consists of four to eight persons and witness is instructed that suspect may or may not be among them. However in this case, suspect can be easily identified even by a mock witness knowing only a short description such as, "Vietnamese male, 50-70 years old."

rank the list of proposed candidates according to the long-term preferences of the user.

More specifically, main contributions of this paper are:

- Proposed and evaluated hybrid personalized recommendation method.
- Dataset of assembled lineups with both positive and negative training examples.

To the best of our knowledge, our work is the first application of recommender systems principles on the lineup assembling task.

2 Item-based Recommendations

2.1 Dataset of Lineup Candidates

Although there are several commercial lineup databases[1], we need to approach carefully while applying such datasets due to the problem of localization. Not only are the racial groups highly different e.g., in North America (where the datasets are mostly based) and Central Europe, but other aspects such as common clothing patterns, haircuts or make up trends vary greatly in different countries and continents. Uunderlined datasets should follow the same localization as the suspect in order to inhibit the bias of detecting strangers or having the incorrect ethnicity in a lineup. We evaluated the proposed methods in the context of the Czech Republic. Although the majority of the population is Caucasian, mostly of Czech, Slovak, Polish and German nationality, there are large Vietnamese and Romany minorities which make lineup assembling more challenging. We collected the dataset of candidate persons from the *wanted and missing persons* application[2] of the Police of the Czech Republic. In total, we collected data about 4,423 missing or wanted males. All records contained a photo, nationality, age and appearance characteristics such as: (facial) hair color and style, eye color, figure shape, tattoos and more. More information about the dataset may be found in [8].

2.2 Item-Based Recommending Strategies for Lineup Assembling

In our previous work [8], we proposed two non-personalized recommending strategies, where the list of proposed candidates is based on the similarity between the suspect and lineup candidates. We use the underlined assumption that the lineup fairness can be approximated through the similarity of the suspect and fillers, i.e. by filling lineups with candidates similar to the suspect, we ensure that lineups remain unbiased.

Content-based Recommendation Strategy (CB-RS) leverages the collected content-based attributes of candidates. We employed the Vector Space Model [3] with binarized features, TF-IDF weighting and cosine similarity. *CB-RS* strategy was intended to be closely similar to the attribute-based searching, which is commonly available in lineup assembling tools.

Recommendation Based on visual features (Visual-RS) leverages the similarity of visual descriptors received from a pre-trained CNN (VGG network for facial recognition problems, VGG-Face [7], in our case). More information is available in the previous work [8].

2.3 Evaluation of Item-Based Recommenders

To make this paper self-contained, let us briefly describe the results of non-personalized recommendation strategies.

The evaluation was based on a user study of domain experts, i.e., forensic technicians, whose task was to select best lineup candidates out of the ones recommended by both techniques. More specifically, 30 persons were selected from the dataset to play the role of suspects. For each suspect, both non-personalized recommendation strategies proposed top-20 candidates that were merged into a single list[3] and displayed together with the suspect to the domain experts. Domain experts selects the most suitable candidates; these were considered as positively preferred. Participants were instructed to maintain lineup fairness principles, they were allowed to produce incomplete lineups if no more suitable candidates were available, or select more candidates if they were equally eligible.

The evaluation was performed by seven forensic technicians from the Czech Republic, with 202 assembled lineups and 800 selected candidates in total. Table 1 illustrates overall results of the user study. One can observe that although *Visual-RS* clearly outperformed *CB-RS*, also the candidates recommended by *CB-RS* were selected quite often. Together with the surprisingly low size of the intersection (1.83%) between the lists of recommended candidates and relatively high level of disagreement among participants on the selected candidates, the results indicate that some merged, personalized strategy is plausible. Furthermore, as the mean rank of selected candidates was decision whether the next list item will be filled by *CB-RS* or *Visual-RS* method.

Table 1: Evaluation results depicting the volume of selected candidates, the differences in volumes of selected candidates (p-value of paired t-test), the level of agreement among participants (Krippendorff's alpha) and the average rank of the selected candidates. Note that candidates proposed by both strategies were excluded from results.

	Selected candidates	P-value	Level of agreement	Average rank
Visual-RS	**466** / 58%	1.2e-8	0.178	8.2
CB-RS	298 / 37%		0.138	8.9

[1] e.g., *http://elineup.org*

[2] *aplikace.policie.cz/patrani-osoby/Vyhledavani.aspx*

[3] The ordering of candidates proposed by each method was maintained, i.e., the randomness was applied on the

relatively high for both methods (8, resp. 9 out of 20), there is a room for some re-ranking approach.

3 Personalized Recommendations

Based on the evaluation of non-personalized, item-based recommending techniques, we hypothesized that the proposed recommendations can be further improved by employing some content-based personalized techniques. We approach this task through state-of-the-art machine learning methods as follows.

Suppose that for arbitrary user u, his/her previous interactions with the system are in the form of triples $F_u: \{r_u(i,j)\}$, where i is the suspect of some previously created lineup, j is a recommended candidate and $r_u = 1$ if j was selected to the lineup and $r_u = 0$ otherwise. Furthermore, both i and j can be represented by three sets of attributes:

- A^{cb} are TF-IDF values of content-based attributes of each object.
- A^{vis} represents the visual descriptor based on the VGG-Face network.
- The union of both sets: $A^{cb} \cup A^{vis}$

Suppose that equations below represents scoring functions of the non-personalized recommending strategies.

$$s_{cb}(i,j) = \frac{1}{1 + \sum_{a \in A^{cb}} |a_i - a_j|} \quad s_{vis}(i,j) = \frac{1}{1 + \sum_{a \in A^{vis}} |a_i - a_j|}$$

Now, let us define a personalized classification / regression task[4] with the train set examples constructed as follows. For each $f \in F_u$, the output variable $y = r$ and the list of dependent variables $\mathbf{x}_A$ are constructed as a subtraction of suspect's and candidate's attributes for a set of attributes A: $\forall a \in A: x_a \coloneqq |a_i - a_j|$.

Given an arbitrary classification method M, the model of user preferences $m_{u,A}$ is trained by applying method M on the per-user train set $\{(\mathbf{x}_A, y)\}$. When the user starts a new lineup task with some new suspect $\bar{\imath}$, the lineup candidates are ranked according to their probability to be selected in the lineup:

$r_j \coloneqq P(r_u(\bar{\imath}, j) = 1 | m_{u,A})$.

We would like to note that such recommendation scenario is quite challenging as we do not have any feedback from the current lineup and need to rely solely on the long-term user preferences (note the relation to the page zero problem or homepage recommendation problem). On the other hand, quite complex learning methods can be used, because the time-span between two consecutive lineup assembling performed by the same forensic technician tends to be rather large.

Following preference learning methods were evaluated[5]:

- Non-personalized similarity based on the L_1 distances (baseline)
- Linear regression (denoted as LM in the evaluation)
- Lasso regression (Lasso)
- Decision tree (Dec. tree)
- Gradient boosted tree (GBT)

As the initial evaluation of the proposed method was only partially successful (machine learning methods were to able significantly improve the baseline only in the case of A^{cb}attribute set), we further proposed a hybrid approach integrating two components:

- Predictions of a selected machine learning method on A^{cb} attribute set.
- Predictions based on a non-personalized L_1 distance metric applied on A^{vis} attribute set.

Both prediction techniques are aggregated via probabilistic sum, i.e., $r_j \coloneqq r_j^{cb} + r_j^{vis} - r_j^{cb} \times r_j^{vis}$. This approach is denoted as *hybrid* in the evaluation.

3.2 Evaluation of Personalized Recommendations

The main goal of the personalized recommendations evaluation is to clarify, whether the long-term user preferences, i.e., collected during some previous lineups assembling, can be utilized to improve the list of recommended candidates for the current lineup.

In order to confirm this hypothesis, we performed an off-line evaluation on the dataset of assembled lineups collected during the evaluation of item-based recommendations. The resulting dataset contained in total 7659 records (800 positive and 6859 negative), i.e., in average 1094 records per user. Proposed methods were evaluated based on the 10-fold cross-validation protocol applied on the lineups. Hyperparameters of the methods were learned via grid-search on an internal leave-one-lineup-out protocol.

For each tested lineup, each recommending method re-ranks objects originally displayed to the forensic technicians according to the computed relevance r_j (selected candidates should appear on top of the list). We measure normalized discounted cumulative gain (nDCG), recall at top-10 and recall at top-5 (rec@10, rec@5 resp.) of the list and report on the average results for all evaluated users and lineups.

Table 2 depicts results of the off-line evaluation. We can observe that both linear model and gradient boosted trees improved over the baseline method in case of the A^{cb} attributes set. Therefore, we evaluated the hybrid approach with both methods. Both hybrid methods outperformed the best baselines w.r.t. nDCG and rec@5 metrics, while GBT hybrid provides the best performance w.r.t. all evaluated metrics.

[4] Please note that although the classification is a natural choice due to the binary output variable, the final output of the method should be ranking of candidates. Thus, we also evaluate several regression-based machine learning methods and in case of classification method, we use positive class probability score as ranking.

[5] We use the methods' implementation from sci-kit package, http://scikit-learn.org.

Table 2: Results of the personalized recommendation methods. Note that A^{vis} based machine learning approaches did not improve the baseline and were omitted for the sake of space.

Method	Attributes	nDCG	rec@10	rec@5
Baseline	A^{cb}	*0.4088*	*0.1796*	*0.0805*
Baseline	A^{vis}	*0.4990*	*0.3837*	*0.1725*
Baseline	$A^{cb} \cup A^{vis}$	*0.4201*	*0.2432*	*0.1090*
LM	A^{cb}	0.4605	0.2949	0.1413
Lasso	A^{cb}	0.3816	0.1255	0.0484
Dec. tree	A^{cb}	0.3842	0.0871	0.0611
GBT	A^{cb}	0.4563	0.2728	0.1451
LM hybrid	$A^{cb} \cup A^{vis}$	0.4995	0.3693	0.2003
GBT hybrid	$A^{cb} \cup A^{vis}$	**0.5205**	**0.3843**	**0.2042**

4 Conclusions

The main aim of this work in progress was to analyze the applicability of recommender systems principles in the problem of photo lineup assembling. Although the photo lineup assembling task is both important and time-consuming task, state-of-the-art tools do not provide intelligent search API beyond simple attribute search and to the best of our knowledge, apart from our work, there are no papers utilizing recommending principles in the lineup assembling task.

After the initial evaluation of item-based recommending algorithms, we proposed several variants of content-based personalized recommending algorithms utilizing long term preferences of the user. The off-line evaluation confirmed that long-term preferences can be used to improve the final ranking of candidates, however, only in case of content-based attributes.

Proposed approaches remained ineffective in the case of visual descriptors, so one direction of our future work is to further analyze this problem and providing solutions suitable also for visual descriptors. Siamese networks merging both content-based and visual descriptors seems particularly suitable for the task. Another option is to use visual descriptors as a base for short-term user preferences, i.e., the ones expressed in the current lineup and refine the recommended objects based on the already selected candidates.

Textual description of the suspect also plays an important role in the lineup assembling, as forensic technicians often tries to select candidates that match mentioned, highly specific, features, e.g., scars, skin defects, specific haircut etc. Another direction of our future work would aim to incorporate searching for these specific features in a *"guided recommendation"* API. Selecting specific regions of interest within the suspect's photo seems to be a suitable initial strategy.

Finally, the long term goal of our work is to move from the recommendation of candidates to the recommendation of assembled lineups and to provide a ready-to-use software for forensic technicians.

Acknowledgments

This work was supported by the Czech grants GAUK-232217 and Q48.

References

[1] Brigham, J.C. 1999. Applied issues in the construction and expert assessment of photo lineups. *Applied Cognitive Psychology*. (1999). DOI:https://doi.org/10.1002/(SICI)1099-0720(199911)13:1+<S73::AID-ACP631>3.3.CO;2-W.

[2] Hu, Y. et al. 2008. Collaborative Filtering for Implicit Feedback Datasets. *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining* (Washington, DC, USA, 2008), 263–272.

[3] Lops, P. et al. 2011. Content-based Recommender Systems: State of the Art and Trends. *Recommender Systems Handbook*. F. Ricci et al., eds. Springer US. 73–105.

[4] MacLin, O.H. et al. 2005. PC_eyewitness and the sequential superiority effect: Computer-based lineup administration. *Law and Human Behavior*. (2005). DOI:https://doi.org/10.1007/s10979-005-3319-5.

[5] Mansour, J.K. et al. 2017. Evaluating lineup fairness: Variations across methods and measures. *Law and Human Behavior*. (2017). DOI:https://doi.org/10.1037/lhb0000203.

[6] Meissner, C.A. and Brigham, J.C. 2001. Thirty Years of Investigating the Own-Race Bias in Memory for Faces: A Meta-Analytic Review. *Psychology, Public Policy, and Law*.

[7] Parkhi, O.M. et al. 2015. Deep Face Recognition. *Procedings of the British Machine Vision Conference 2015* (2015).

[8] Peska, L. and Trojanova, H. 2017. Towards recommender systems for police photo lineup. *ACM International Conference Proceeding Series* (2017).

[9] Shapiro, P.N. and Penrod, S. 1986. Meta-Analysis of Facial Identification Studies. *Psychological Bulletin*.

[10] Steblay, N. et al. 2003. Eyewitness Accuracy Rates in Police Showup and Lineup Presentations: A Meta-Analytic Comparison. *Law and Human Behavior* (2003).

[11] Valentine, T.R. et al. 2007. How can psychological science enhance the effectiveness of identification procedures? An international comparison. *Public Interest Law Reporter*. (2007). DOI:https://doi.org/10.1017/CBO9781107415324.004.

S. Krajči (ed.): ITAT 2018 Proceedings, pp. 161–167
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Jakub Šiller and Jaroslav Kuchař

Framework for Distributed Computing on the Web

Jakub Šiller and Jaroslav Kuchař

Web Intelligence Research Group, Faculty of Information Technology
Czech Technical University, Thákurova 9, 160 00, Prague 6, Czech Republic
{sillejak|jaroslav.kuchar}@fit.cvut.cz

Abstract: This work is a brief summary of a master thesis that focuses on design and implementation of a framework that uses computers of website visitors as computing nodes through web browsers. It contains an analysis of the Web environment, summarization of previous approaches and projects, design and implementation of the framework. The work describes the solution of computing node failure, reaction to slow computing node, possibilities of controlling the load of the framework on a website visitor's computer, strategies for work distribution and security of the framework. At the end of the work, the experiment results and proposal of improvements are listed.

1 Introduction

Nowadays, web technologies and web services are an inseparable part of human life. We are using web browser for communication, entertainment, shopping and a many other activities. A lot of web users have powerful devices but they don't use their full power often. Thus, in the world, there is huge computing potential that is idle. Framework presented in this work is able to utilize this computing power.

In section 2 there is a summarization of previous works and current project in the field of distributed computation in web browsers. Analysis of web environment and framework requirements are content of section 3. Section 4 is focused on design of the framework. Finally, experiment results are presented in section 5.

2 Related works

Utilizing idle power via the Internet to create distributed computer is not a new idea. According to [1] the first project about this topic was *Great Internet Mersenne Prime Search* [2] that started in 1996. A few years later projects *SETI@Home* [3] that was focused on the Search for Extraterrestrial Intelligence and *Folding@home* [4] that was using computers of volunteers for medical research was launched. After that *BOINC (Berkeley Open Infrastructure for Network Computing)* [5] was created. It is probably the biggest and the best known platform for volunteer computing. All of mentioned projects are still alive. In order to join a computation in these projects a volunteer have to install some additional software.

Since 2007 several works related to utilizing computation power using web browser have been published. They were focused on various types of computing tasks. For instance projects [6], [7], [8] and [9] were using web browsers for simulated evolution. Work [10] was focused on image processing. Machine learning in web browsers was the topic of [11]. Authors of [12] created web search engine using web browsers. In [13] and [14] map reduce frameworks were presented. There were also works that were focused on creating general framework: [15], [16], [17]. Architecture of most of presented works was client-server.

Developers of commercial project *Computes* are trying to create distributed decentralized supercomputer from all kind of devices.

There are also several commercial tools for mining altcoins.

Published works are usually just proof of concept and authors don't deal with every aspect of systems. For instance security is often omitted. The main contribution of this work is to create complex framework that could be deployed.

3 Analysis

The web has several major characteristics. One of them is *diversity*. Web users are using various browsers of various versions. Each user has different device, different connection speed etc.

Another strong aspect of the Web is *dynamics*. Web technologies are still evolving and continuously new technologies are emerging. Also web browsing is very dynamic. According to [18] users often stay on a page just for 10-20 seconds.

The Web is *free and open*. Everyone can join and publish and consume data.

And the last but not least aspect is its' *enormous size*. [19] and [20] states that the Web has more than 3.7 billions users and this number is getting bigger every year.

These aspects of the Web have impact on the framework requirements. Dynamic browsing will cause frequent computing node failures. The framework have to be able to detect failure and solve the situation. The framework also should be able to work with computing nodes of different performance. Security mechanisms should be involved on server side and on client side as well. Framework also have to ensure correctness and reliability of tasks results.

4 Design

4.1 Computational model

The framework is focused on types of tasks in which server sends work to client, client processes the work and sends the result back to the server. There is no communication between clients and no communication between client and server regarding the computation but distributing works and receiving results. Framework user can define dividing and merging functions. In case that a task has too big data the framework will use dividing function in order to automatically recursively divide the data and create partial subtasks called works. Works are then distributed to the client. Results of works from clients are then merged by framework using merging function. Described model is shown in figure 1.

In order to compute a tasks there will be a lot of messages between clients and the server. Speed of communication depends on connection quality. It might be time consuming. Therefore, the framework is more suitable for computing intensive tasks rather than data intensive tasks.

4.2 Users

Users of the framework are divided into two groups. Users from the first group are creators of task prototypes. A task prototype consists of definition of code that should be executed in client and optional dividing and merging function. This group of users should know the framework - its' advantages, disadvantages and some technical details in order to create effective code for the framework.

The second group are common users. They use framework for computations. They don't need to know anything about the framework but the id of task prototype they want to use.

This division has several reasons. The first of them is security. We can assume that number of task prototypes creators will be much smaller than number of other users. Therefore it might be relatively easy to make sure that creators of task prototypes are trustworthy. And then we can assume that their code is probably trustworthy as well. Another reason for the division is quality of code. If there are users who are focusing on creating code for the framework there is high probability that the code will be efficient, without bugs and there will be no task prototypes for tasks that are not suitable for the framework. Moreover common users don't need to know anything about the framework.

4.3 Framework in a nutshell

Basic architecture of the framework is client-server. The server consists of two main parts - *ProgrammerServer* and *VolunteerServer*. *ProgrammerServer* is responsible for communication with users of the framework. *VolunteerServer* is the main part of the framework. It is responsible for communication with clients, processing tasks, distributing works to clients and processing results.

The framework utilizes replication and majority voting in order to ensure correctness and reliability of works results. User can for each task specify the replication factor. In each work object server holds information that indicates to how many more clients the work should be distributed in order to reach the replication factor. This information is in attribute *remaining*.

In order to compute a task a user sends data and id of task prototype to the server. Server insert the task to the task queue. The server holds collection of works that are currently being distributed to clients. If there is enough place for more works server prepare another task from the queue for distribution. Preparation of a task for distribution consists mainly of dividing task's data. When a client sends work request the server response with several works. Works that are returned to the client have attribute *remaining* greater then zero. After assigning a work to a client the attribute is decremented. Number of works that are returned to the client depends on strategy that is used. Strategies implemented in the framework are described in following section. When a client receives works it starts to process them. The client doesn't wait until all works are done but each result is sent back to the server as soon as it is available. This is shown in figure 3.

When the server received results from all works it merge them to the result of the task. The task's result is stored in database. When the user send request for the result to the server it responds with the result from the database.

When a client processes all assigned works it sends new work request to the server.

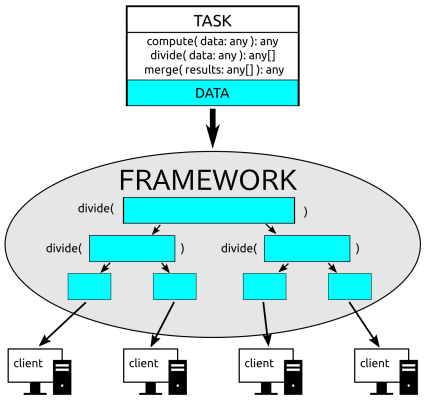
Strategies for work distribution There are several strategies in the framework for determining number of works that should be returned to a client.

The most easy one is a strategy that returns fixed number of works.

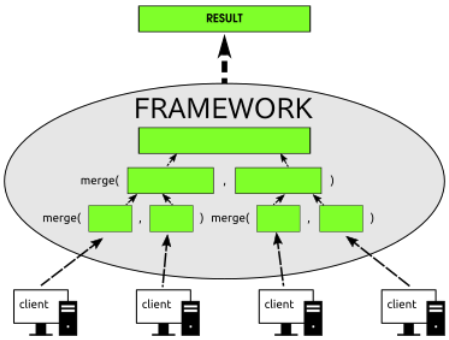
Another strategy is based on fixed sum of works sizes. The framework uses statistics to estimate maximum number of works so that sum of their sizes is less then configured threshold.

The most complex strategy uses time elapsed from start of a client session. According to [18] a distribution of session duration follows Weibull distribution with negative aging. That means that at the begging of a web page visit the probability that user will leave the page is very high and it decrease over time. Therefore this strategy increase number of works that are sent to the client accordingly to the session duration.

Modes of work distribution Work distribution can run in two modes. In the first mode data with full work code are sent to the client. In the second mode data along code identifier are sent. In this mode client have to make another request to the server in order to get the code. However, in this mode it is possible to cache the code in the



(a) Dividing and distributing phase



(b) Receiving and merging phase

Figure 1: Visual example of the computation model of the framework

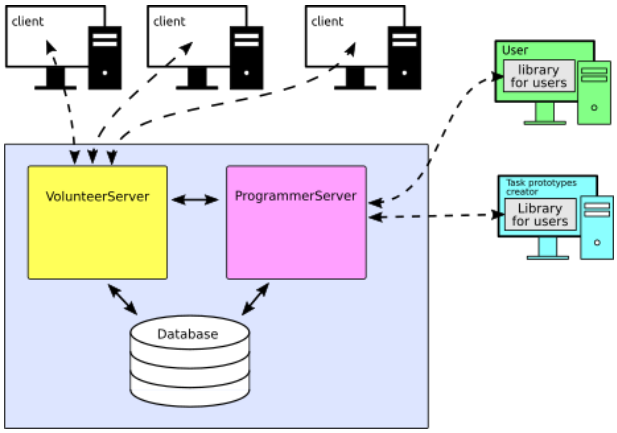


Figure 2: Base architecture of the framework.

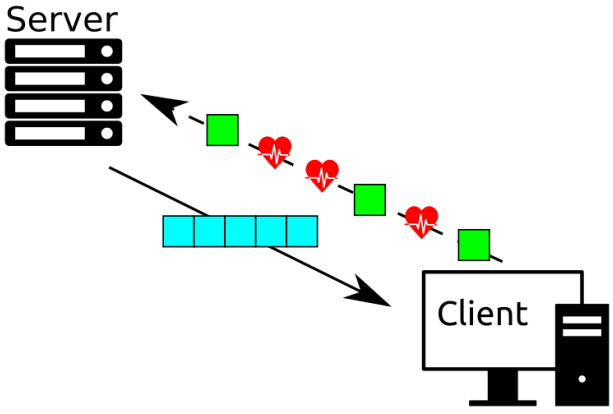


Figure 3: Simplified process of distributing works to the client and subsequent sending results back to the server. During computation of a work heartbeats are sent to the server.

browser and in intermediate network devices. Therefore the amount of data on wires can be decreased. This mode is efficient only if a client or a group of clients are performing tasks of a single or a few task prototypes. In other cases it might be inefficient because of more requests.

4.4 Computing node failure

In order to detect computing node failure the framework holds in each session object time of last access. It is updated with each request from the session. During computation a client code running in a session periodically sends empty request to the server - a heartbeat - so the server is informed the session is still alive as is shown in figure 3.

A server module *DeadSessionCollector* periodically collects dead sessions. A session is considered to be dead if time elapsed from the last access is higher than a configured threshold. When a session die the framework increments attribute *remaining* of all unprocessed works that were assigned to the session. So the work will be assigned again to some session.

4.5 Slow computing node

Because of different performance of web user devices, different connection speed and different utilization of the device by it's user it may happen that computation of a work would last on one client much longer than computation of the same work on other client. Thus, when some client is processing some work too long it may be efficient to assign the work again to another client.

The framework's module *LongRunningSessionCollector* periodically checks whether there is a slow client for some work. A client is considered too slow for a work if

the time elapsed since the work was assigned to the client is several times longer than average time of computation of the work on other clients. If a client is marked as slow for a work the framework increment attribute *remaining* of the work.

4.6 Client side

At client side the framework is using Web Worker technology [21]. A web browser create new thread for each Web Worker object. Therefore experience of browsing a web page should not be affected by the framework. The framework creates several Web Worker objects in which works are processed. When a client receive works it put them in a queue. The Web Worker objects are taking objects from the queue and processing them. As soon as a Web Worker object computes a work the result is sent to the server. This is shown in figure 4.

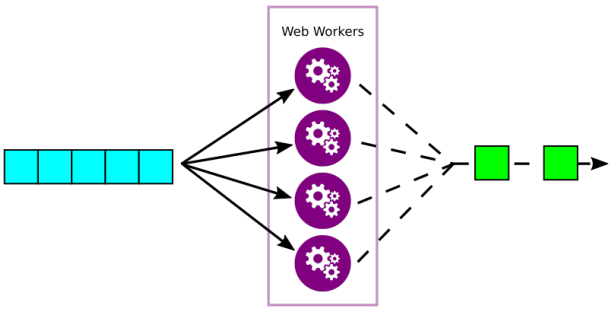


Figure 4: Illustration of processing works assigned to a client on multiple Web Worker objects

For computation of a work the framework is able to use new client-side web technologies asm.js [22] and WebAsembly [23] instead of JavaScript [24]. The creator of task prototype can implement working function in C++ which is then at the server compiled into asm.js and WebAssembly modules.

4.7 Controlling of client utilization

There are two ways how the framework is able to control stressing of client's device. The first one is controlling the number of Web Workers that are used. The second option is throttling. JavaScript code can not control stressing of CPU at particular time but it is possible to control stressing from long-term view. For instance if x seconds is processor fully stressed by the framework and then x seconds is idle we can say that the framework stressed CPU by 50% during $2x$ seconds. The configuration attribute *throttleFactor* holds information how many times the duration of the last computation the server has to wait before it could again assign a work to a client. In other words it is inverted value to the desired CPU usage.

4.8 Security

The framework contains several security mechanisms. The server accepts user request only with valid API keys. List of API keys is in configuration of the framework. If a request doesn't contain valid API key the server responds with HTTP code 403 forbidden.

The framework executes user code on server-side and also on client-side. The framework must ensure that the code will cause no harm to server or client's device. Therefore, user code is executed in a sandbox. At the server-side VM2 module [25] is used. At the client-side a sandbox is implemented as a white-list of allowed function and objects. Every other function or object is disallowed.

Communication between a client and the server is encrypted. This decrease probability that content of messages is changed by malicious third-party.

As was mentioned earlier in order to ensure correctness an reliability of results majority voting is involved. A user can specify replication factor but he or she should be aware that probability that data are correct is never 100%.

Securing data is complicated topic. The purpose of web browsers is to serve data to its user so the framework can not hide data from the user of the browser it is running in. Therefore the only way how to secure the data is computing on encrypted data. There are mathematical models that enables it. Some of them are described in [26]. There is no need to change the framework in order to compute on encrypted data - it is responsibility of task prototypes implementation. Computing on encrypted data can also ensure 100% probability that result is correct excluding bugs in the task prototype.

5 Experiments

Experiments were performed on 60 computers in classrooms at FIT CTU. Configuration of computers is in table 1. Test task was naive algorithm for computing determinant of a matrix. Maximum size of a matrix that was sent to clients was 11×11. If the matrix was bigger it was recursively divided to matrices of size 11×11.

Classroom	CPU Model	size of RAM
T9-350	Intel® Core™ i5-6500 CPU @ 3.20GHz	16 GB
T9-351	Intel® Core™ i5-3470 CPU @ 3.20GHz	8 GB
T9-303	Intel® Core™ i5-3470 CPU @ 3.20GHz	8 GB
T9-349	Intel® Core™ i5-4570S CPU @ 2.90GHz	8 GB
server	Intel® Core™i5-2410M CPU @ 2.30GHz	6 GB

Table 1: Configuration of test computers

Tests were performed in the mode in which just identifier of a code is sent to the client. Replication factor was set to 3. The framework was using all CPU cores of the client devices. Test environment refreshed test web page at a client after randomly chosen time from interval $[0, maxDuration]$ where *maxDuration* is parameter. Test Settings of earlier mentioned parameters are in table 2.

Parameter	Value
deadSessionCollector:	
interval	2 sec
deadTime	10 sec
LongRunningSessionCollector:	
interval	5 sec
factor	5
Other framework settings:	
throttleFactor	0
heartBeatInterval	5 sec
test environment settings:	
maxDuration	960 sec

Table 2: Test setting of mentioned parameters

The purpose of the first experiment was to test how would change computation time of the task from user point of view with increasing number of clients. In figure 5 we can see that with increasing number of clients computation time is decreasing. During tests in classrooms T9-350, T9-351 and T9-303 speeding up stops around number 30. That probably happened because computers in T9-350 were more powerful that others (shown in figure 7) so the framework considered the others to be slow and started to assign one work to more clients that was necessary. This is shown in figure 6. Also there was a mistake in the configuration. Interval of sending heartbeat was equal to the time after which a session was considered as dead. After some changes in configuration the computation becomes again a little bit faster.

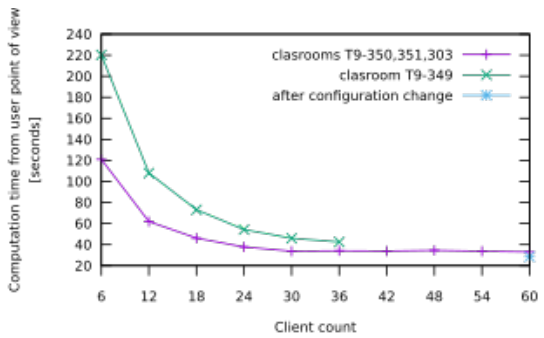


Figure 5: Influence of number of clients to computation time from user point of view.

The second experiment tests how size of a matrix would affect computation time from user point of view. It also compares computation time of the task using framework and computation time using local computation that was

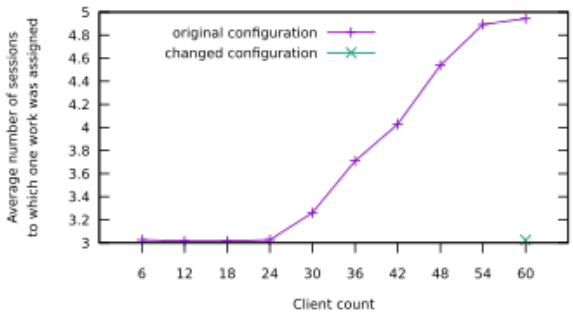


Figure 6: Influence of number of clients to average number of sessions to which one work was assigned

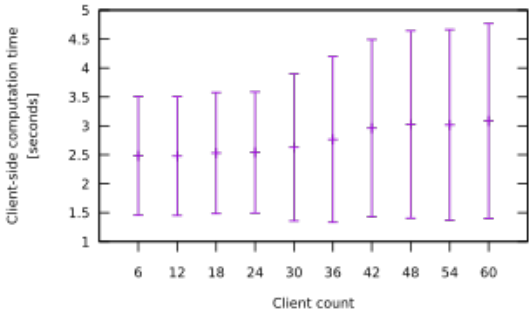


Figure 7: Statistics of a computation time of one matrix at client-side. For each number of client on axis X mean and mean $\pm$ standard deviation of computation time at client-side is shown

implemented in C++ using OpenMP [27] in order to utilize all CPU cores of a computer. Local computation was executed on the server's computer. In figure 8 we can see that for small matrices the local computation was more efficient but for big matrices it was more efficient to use the framework.

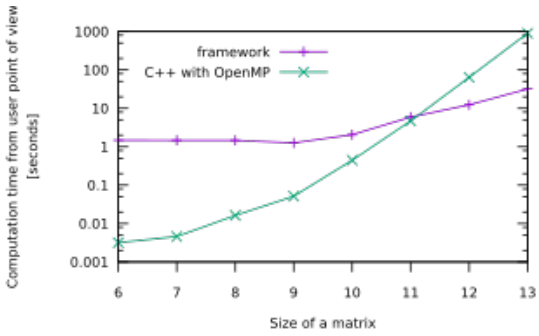


Figure 8: Influence of matrix size to computation time from user point of view.

The third experiment tests influence of *throttleFactor* on the computation time from user point of view. In figure 9 we can see that there is linear dependency. This test prove that influence of *throttleFactor* is expected and predictable.

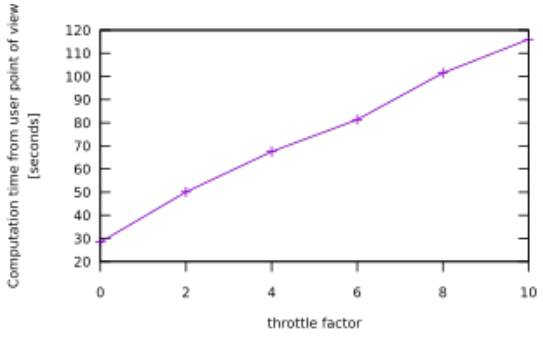


Figure 9: Influence of attribute *throttleFactor* to computation time from user point of view.

The last experiments tests how computation time from user point of view is changing when maximal session duration is changing. In figure 10 we can see that with increasing session duration computation time is decreasing as it was expected.

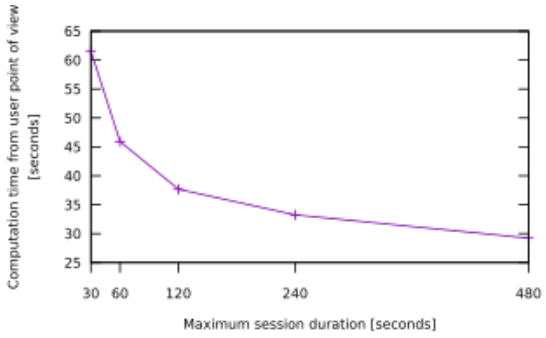


Figure 10: Influence of maximal session duration (attribute *maxDuration* of test environment) to computation time from user point of view.

6 Future work

Experiments have shown that framework is useful and it may be deployed. However there are still some limits and drawbacks of the implementation that should be solved. For instance database queries can be optimized or new strategies for distributing works can be implemented.

The framework can be also extended in several ways. For instance GUI and monitoring extension would be very practical. There might be client, user and web sites administration. Extensions of the framework may lead to computing platform with market that would act similar as application markets.

7 Motivation for joining computations

There can be several reasons for a web user to join the framework but probably the most likely situation is this: A web page could profit from involving it's visitors to the framework. So the web page could offer a discount of their services to visitors that allow joining to the framework.

Another use case could be in a company that have a lot of computers for it's employees. In this case a proxy could inject web pages with framework's client-side code and so create company's big distributed computer with minimum additional costs.

8 Conclusion

In this work a framework for distributed computation using web browsers is presented. It contains mechanisms that solve computing node failure and reaction to a slow computing node. It describes strategies for distributing works within clients and modes in which the distribution can be done. The work deals with controlling of client's device stressing. Security mechanisms are described as well. Experiment results that are presented have shown that the framework is useful and deployable. The future of the framework may be a computational platform with market of task prototypes and computation power.

Acknowledgments

This research was supported by Faculty of Informatics, Czech Technical University in Prague.

References

[1] Miller, D. M.: The Online Community Grid Volunteer Grid Computing with the Web Browser. Georgia Institute of Technology, 2008. Available on: `https://smartech.gatech.edu/handle/1853/33477`

[2] Mersenne Research, Inc.: Great Internet Mersenne Prime Search. [cit. 30.4.2018]. Available on: `https://www.mersenne.org`

[3] University of California : About SETI@home. [cit. 30.4.2018]. Available on: `https://setiathome.berkeley.edu/sah_about.php`

[4] Pande Lab: Folding@home. [cit. 30.4.2018]. Available on: `http://folding.stanford.edu`

[5] Anderson, D. P.: BOINC: a system for public-resource computing and storage. In *Fifth IEEE/ACM International Workshop on Grid Computing*, Nov 2004, ISSN 1550-5510, s. 4–10, doi:10.1109/GRID.2004.14.

[6] Merelo, J. J.; García, A. M.; Laredo, J. L. J.; aj.: Browser-based Distributed Evolutionary Computation: Performance and Scaling Behavior. In *Proceedings of the 9th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '07, New York, NY, USA: ACM, 2007, ISBN 978-1-59593-698-1, s. 2851–2858, doi:10.1145/1274000.1274083. Available on: `http://doi.acm.org/10.1145/1274000.1274083`

[7] Klein, J.; Spector, L.: Unwitting Distributed Genetic Programming via Asynchronous JavaScript and XML. In *Proceedings of the 9th Annual Conference on Genetic and*

Evolutionary Computation, GECCO '07, New York, NY, USA: ACM, 2007, ISBN 978-1-59593-697-4, s. 1628–1635, doi:10.1145/1276958.1277282. Available on: http://doi.acm.org/10.1145/1276958.1277282

[8] Merelo-Guervos, J. J.; Castillo, P. A.; Laredo, J. L. J.; aj.: Asynchronous distributed genetic algorithms with Javascript and JSON. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, June 2008, ISSN 1089-778X, s. 1372–1379, doi:10.1109/CEC.2008.4630973.

[9] Duda, J.; Dlubacz, W.: Distributed Evolutionary Computing System Based on Web Browsers with Javascript. In *Proceedings of the 11th International Conference on Applied Parallel and Scientific Computing*, PARA'12, Berlin, Heidelberg: Springer-Verlag, 2013, ISBN 978-3-642-36802-8, s. 183–191, doi:10.1007/978-3-642-36803-5_13. Available on: http://dx.doi.org/10.1007/978-3-642-36803-5_13

[10] Zorrilla, M.; Martin, A.; Tamayo, I.; aj.: Web Browser-Based Social Distributed Computing Platform Applied to Image Analysis. In *2013 International Conference on Cloud and Green Computing*, Sept 2013, s. 389–396, doi:10.1109/CGC.2013.68.

[11] Meeds, E.; Hendriks, R.; al Faraby, S.; aj.: MLitB: Machine Learning in the Browser. *CoRR*, ročník abs/1412.2432, 2014, 1412.2432. Available on: http://arxiv.org/abs/1412.2432

[12] Turek, W.; Nawarecki, E.; Dobrowolski, G.; aj.: WEB PAGES CONTENT ANALYSIS USING BROWSER-BASED VOLUNTEER COMPUTING. *Computer Science*, ročník 14, č. 2, 2013: str. 215, ISSN 2300-7036. Available on: https://journals.agh.edu.pl/csci/article/view/278

[13] Pan, Y.; White, J.; Sun, Y.; aj.: Gray Computing: An Analysis of Computing with Background JavaScript Tasks. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, ročník 1, May 2015, ISSN 0270-5257, s. 167–177, doi:10.1109/ICSE.2015.38.

[14] Ryza, S.; Wall, T.: MRJS: A JavaScript MapReduce Framework for Web Browsers. 2010. Available on: http://static.cs.brown.edu/courses/csci2950-u/f11/papers/mrjs.pdf

[15] Cushing, R.; Putra, G. H. H.; Koulouzis, S.; aj.: Distributed Computing on an Ensemble of Browsers. *IEEE Internet Computing*, ročník 17, č. 5, Sept 2013: s. 54–61, ISSN 1089-7801, doi:10.1109/MIC.2013.3.

[16] Wilkinson, S. R.; Almeida, J. S.: QMachine: commodity supercomputing in web browsers. *BMC Bioinformatics*, ročník 15, č. 1, Jun 2014: str. 176, ISSN 1471-2105, doi:10.1186/1471-2105-15-176. Available on: https://doi.org/10.1186/1471-2105-15-176

[17] Fabisiak, T.; Danilecki, A.: Browser-based Harnessing of Voluntary Computational Power. ročník 42, 03 2017. Available on: https://www.degruyter.com/downloadpdf/j/fcds.2017.42.issue-1/fcds-2017-0001/fcds-2017-0001.pdf

[18] Nielsen, J.: How Long Do Users Stay on Web Pages? [cit. 30.4.2018]. Available on: https://www.nngroup.com/articles/how-long-do-users-stay-on-web-pages/

[19] Liedke, L.: 100+ Internet Stats and Facts for 2018. [cit. 30.4.2018]. Available on: https://www.websitehostingrating.com/internet-statistics-facts-2018/

[20] Stevens, J.: Internet Stats & Facts for 2017. [cit. 30.4.2018]. Available on: https://hostingfacts.com/internet-facts-stats-2016/

[21] Mozilla and individual contributors: MDN web docs: Using Web Workers. [cit. 30.4.2018]. Available on: https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API/Using_web_workers

[22] Herman, D.; Wagner, L.; Zakai, A.: asm.js: Working Draft. [cit. 30.4.2018]. Available on: http://asmjs.org/spec/latest/

[23] WebAssembly. [cit. 30.4.2018]. Available on: http://webassembly.org/

[24] Mozilla and individual contributors: MDN web docs: JavaScript reference. [cit. 30.4.2018]. Available on: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference

[25] Šimek, P.: VM2. [cit. 30.4.2018]. Available on: https://www.npmjs.com/package/vm2

[26] Vaikuntanathan, V.: How to Compute on Encrypted Data. In *Progress in Cryptology - INDOCRYPT 2012*, editace S. Galbraith; M. Nandi, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, ISBN 978-3-642-34931-7, s. 1–15.

[27] OpenMP ARB: OpenMP. [cit. 30.4.2018]. Available on: http://www.openmp.org/

Combinatorics on words

The workshop **Combinatorics on words** (and related topics) is an informal continuation of the successful and productive series of meetings Česko-Slovenská Mela, held annually at the end of September from 2012 - 2015 in Telč.

The workshop is a meeting of researchers and students working primarily in the areas of

- Combinatorics on words,
- Automata and languages,
- Enumeration systems,
- Stringology and Arbology,
- Recursive relations,
- Data compression.

It is intended to be a meeting place for different working groups mostly from the Czech Republic and Slovakia, but an invitation to participate was extended to colleagues outside these two countries, therefore the language of the meeting is English. Presentations and papers for the workshop were solicited. All papers submitted were reviewed by two independent reviewers. The format of the workshop consists of sections where new results and work in progress are presented, as well as sections (and that is the main focus of our meeting) where open problems and common projects are discussed, proposed, and worked on.

The organizers of the workshop are grateful to the organizing and program committees of ITAT for the chance to renew the series of conferences within the frame of ITAT and for all their help.

Tatiana Jajcayová, (Comenius University, Bratislava, Slovakia)
Karel Klouda (Czech Technical University, Prague)
Edita Pelantová (Czech Technical University, Prague)
Štepán Starosta (Czech Technical University, Prague)
Workshop organizers

CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073,

Palindromy pohledem kombinatoriky na slovech

Petr Ambrož

ČVUT

Abstract: Přednáška se zabývá kombinatorickými vlastnostmi palindromů, tj. slov stejných ať je čteme zepředu nebo zezadu. V poslední době bylo formulováno několik různých domněnek o množině palindromů ve faktorových jazycích. Některé z nich jsou motivovány aplikacemi v jiných vědních oblastech, jako jsou fyzika pevných látek nebo genetika. Podáme přehled známých částečných výsledků o platnosti zmíněných domněnek a ukážeme jejich vzájemnou propojenost.

Petr Ambrož získal doktorát v Informatice na Université Paris 7 a v matematickém inženýrství na Fakultě jaderné a fyzikálně inženýrské Českého vysokého učení technického v Praze, kde dnes působí na Katedře matematiky. Zabývá se kombinatorikou na slovech a nestandardními numeračními systémy a je organizátorem nejvýznamnějších konferencí v těchto oborech (WORDS 2011, Numeration 2008, 2016) .

S. Krajči (ed.): ITAT 2018 Proceedings, pp. 171–175
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Martin Nehéz

Probabilistic Analysis of Graphlet Frequency Distribution in Sparse E-R Random Graphs

Martin Nehéz

Institute of Information Engineering, Automation and Mathematics,
Faculty of Chemical and Food Technology
Slovak University of Technology in Bratislava
Slovak Republic
martin.nehez@stuba.sk
WWW home page: https://www.kirp.chtf.stuba.sk/index.php?menu=2& show_id=3& person_id=300011

Abstract: Frequency counting of graphlets (i.e. small connected induced subgraphs) is a prominent experimental approach to network analysis which became favorable in bioinformatics. In this paper, we use the probabilistic method for graphlets counting. We show that it is possible to enumerate the graphlets (or isolated graphlets) in sparse Erdős-Rényi graph model analytically. Obtained frequencies are exploited to estimate bounds on domination number in the above mentioned random graph model.

1 Introduction

Network science which is focused on modeling and analysis of real-world networks became a significant research area in last two decades. Instances of networks under study come from many fields of human activities, e.g. electrical engineering, transportation, social science, biology, medicine, etc. Currently, a frequently used algorithmic-experimental method for similarity detection and comparison of protein-protein interaction networks (shortly PPI networks) was invented in bioinformatics by N. Pržulj et al. [16]. It is based on frequency analysis of small connected induced subgraphs (called graphlets) occurring in networks to be compared. Such an approach was used to show e.g., that yeast PPI networks are structurally closer to geometric random graphs than to scale-free or Erdős-Rényi random graphs [16].

Frequency counting of graphlets is a non-trivial algorithmic problem which is intensively studied both from theoretical and application point of view. One of the most powerful softwares used for this purpose is currently the Orbit Counting Algorithm - ORCA [8, 9]. Roughly speaking, ORCA writes numbers of all graphlets (with $2-5$ nodes) which occurred in a given input network on output. This part of the graphlet-based analysis is the most difficult since doing it without a computer program is unrealistic even for relatively small networks. Due to high computational complexity of the graphlet enumeration the number of their nodes is restricted to at most 4 in some current softwares.

In this paper, we show that the problem of frequency counting of graphlets can be solved analytically in Erdős-Rényi random graph model by probabilistic methods. Some previous results regarding random graph theory [1, 3, 10, 11, 17] are relied on this purpose. Our result is significant especially from the complexity point of view because it might lead, in some cases, to elimination of high requirements on computational resources needed for graphlet frequency analysis. It means that instead of computer-based graphlet frequency enumeration in Erdős-Rényi random graphs one may use analytical formulas.

The organization of the paper is as follows. Sect. 2 contains definitions and preliminary facts. The threshold functions for presence of graphlets in random graphs are derived in Sect. 3. Average counts of graphlets are expressed in the same section as well. Average counts of isolated graphlets and corresponding estimations in random graphs with edge probability $p = c/n$ are determined in Sect. 4. Application of these results to estimation of the domination number in sparse random graphs is outlined in Sect. 5. Possible direction for future research are discussed in the last section.

2 Definitions and Preliminaries

2.1 Fundamentals

The asymptotic notation such as o, O, Θ is used in the usual way [10], nevertheless, the most important notions are listed below.

$$O(g(n)) = \{\ f(n) \mid \exists c, n_0 > 0\ \forall n \geq n_0\ |f(n)| \leq c|g(n)|\ \}$$

$$\Omega(g(n)) = \{\ f(n) \mid \exists c, n_0 > 0\ \forall n \geq n_0\ |f(n)| \geq c|g(n)|\ \}$$

$$o(g(n)) = \{f(n) \mid \forall \varepsilon > 0\ \exists n_0 > 0\ \forall n \geq n_0\ |f(n)| \leq \varepsilon |g(n)|\}$$

$$f(n) = \Theta(g(n))) \ \Leftrightarrow\ f(n) = O(g(n)) \wedge f(n) = \Omega(g(n))$$

Moreover, for two sequences (or equivalently functions) $a = (a_n)_{n=0}^{\infty}$ and $b = (b_n)_{n=0}^{\infty}$, we will write $a_n \ll b_n$ if $a_n \geq 0$ and $a_n = o(b_n)$.

Throughout this paper, all graphs are simple, undirected and without weights. Standard notions of the graph theory are used without definitions or further comments. We address [6, 7] for references however, the usage of some symbols is mentioned bellow. Let $G = (V, E)$ be a graph with a nonempty finite set of *vertices* $V(G)$ (or *nodes*) and

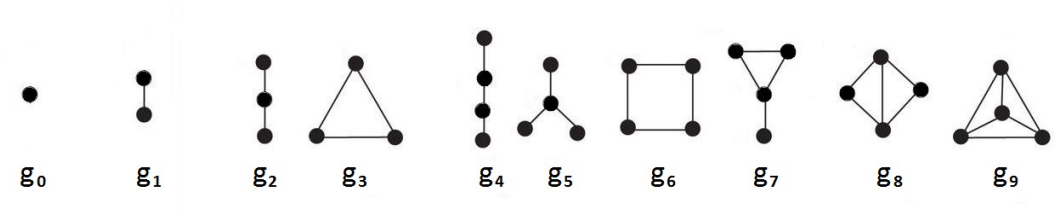


Figure 1: Graphlets $g_0, g_1, \ldots, g_9$.

a finite set of *edges* $E(G)$. Usually $|V(G)| = n$, where $|.|$ stands for the cardinality of a given set. For a vertex $v \in V(G)$, its degree (the number of adjacent vertices of v) is denoted by $\deg(v)$. The maximum (minimum) degree of a graph G is denoted by Δ (δ). Let $G = (V,E)$ be a graph and let $S \subseteq V(G)$, an *induced subgraph* $G[S]$ is the graph whose vertex set is S and the edge set of $G[S]$ consists of all edges whose endpoints are both in S. A graph is said to be *connected* if there is a path joining every pair of its vertices. A *disconnected* graph is not connected. A *component* is a maximal connected subgraph. It is also referred to as *connected component* or *isolated component*. Note that each disconnected graph consists of at least two different components.

Let $G = (V,E)$ be a graph, a *graphlet* is a connected induced subgraph of G with at most 5 vertices. Two graphlets are same if there exists an isomorphism such that it maps one graphlet to the other one. (Two different occurrences of the same graphlet are usually referred to as its copies.) In this paper, only graphlets with at most 4 vertices are considered. Their ordering (and indexing) is shown in Fig. 1. It means that the empty graph (a single vertex) is denoted by g_0, etc. The last graphlet in Fig. 1 (i.e. g_9) is the clique K_4.

2.2 Random Graphs

The *Erdős-Rényi random graph model* (shortly *E-R model*) [3] can be introduced as follows.

Let n be a positive integer and let $p \in \mathbb{R}$ be a constant such that $0 < p < 1$. Consider that for n labeled vertices of $V(G)$, each unordered pair of vertices introduces one *slot* available for an edge. Clearly, the total number of slots is $\binom{n}{2}$. Each edge exists in G independently and with the probability p, thus $\Pr[\{u,v\} \in E(G)] = p$, for all $u,v \in V(G)$.

Given n as above, let $(\Omega, \mathbb{F}, \Pr)$ be a probability space where the sample space Ω consists of all (labeled) graphs G of order n and let $\mathbb{F} \subseteq 2^{\Omega}$ be a set of events. If G has $|E(G)|$ edges, $0 \leq |E(G)| \leq \binom{n}{2}$, then the probability of obtaining G as a result of random edge generation process is given by:

$$\Pr[G] = p^{|E(G)|}(1-p)^{\binom{n}{2}-|E(G)|} . \qquad (1)$$

The probability space $(\Omega, \mathbb{F}, \Pr)$ is denoted by $\mathbb{G}(n,p)$ or $G_{n,p}$ and called *the probability space of all random graphs of order n* or *E-R random graph model*.

A statement about a random graph from $\mathbb{G}(n,p)$ is said to hold *asymptotically almost surely (a.a.s.)* if it holds with probability approaching 1 as $n \to \infty$. A graph G with n vertices is said to be *dense* if it has $\Theta(n^2)$ edges (i.e., asymptotically equal to n^2) and G is said to be *sparse* if it has $o(n^2)$ edges (i.e., asymptotically less than n^2).

If one considers p to be a non-zero constant, then random graphs have $pn(n-1)/2 = \Theta(n^2)$ edges, hence they are dense a.a.s. On the other hand, there are such choices of p that random graphs are sparse. E.g. if $p = p(n)$ is a decreasing function on n such as $p = n^{-\varepsilon}$ for any constant $\varepsilon > 0$ then random graphs have $\Theta(n^{2-\varepsilon})$ edges and they are sparse a.a.s. In particular, if $p = c/n = \Theta(n^{-1})$ for any constant $c > 0$, then random graphs have a linear number of edges (observe that $\varepsilon = 1$, hence random graphs are sparse a.a.s.) and, in terms of edge set cardinality, they are similar to real-world networks. On the other hand, some structural properties of sparse random graphs are usually far from real-world networks [2, 16]. Many works deal with structural properties of real networks and quite realistic models are currently represented by e.g. scale-free or random geometric networks [2, 8, 16].

2.3 Monotonicity and Threshold Functions

In physics, a phase transition is the transformation of a thermodynamic system from one phase to another. During such a transformation, some physical properties of the system change discontinuously. An example is freezing (or boiling) water or the emergence of superconductivity in certain metals when cooled below the critical temperature. In all phase transitions, there exists a value of a certain quantity (often temperature) in which the physical properties in question change.[1] Such a value is said to be a *critical point*.

Similar behavior was also observed in random graphs (for the first time in [3]). The critical points in thermodynamics have their counterparts in random graphs: they are thresholds functions.

[1] The character of such a change is similar to a jump discontinuity function.

Let K_n be a clique with $n > 0$ vertices. Let 2^{K_n} denote the power set of all spanning subgraphs of K_n. Let G_1, G_2 be spanning subgraphs of K_n and let $G_1 \subseteq G_2$ denote that $E(G_1) \subseteq E(G_2)$. Let $Q \subseteq 2^{K_n}$ be a family of subsets and note that Q contains spanning subgraphs of K_n. A family of graphs Q is said to be *increasing* if $G_1 \subseteq G_2$ and $G_1 \in Q$ imply that $G_2 \in Q$. A family of graphs Q' is *decreasing* if $2^{K_n} \setminus Q'$ is increasing. A family which is either increasing or decreasing is called *monotone*. A family of graphs $\mathscr{G}$ is said to be *closed under isomorphism* if for all $G \in \mathscr{G}$ and $H \cong G$ implies that $H \in \mathscr{G}$. If a family of graphs from 2^{K_n} is closed under isomorphism, it can be identified with a graph property.

Threshold functions are usually defined for monotone properties. In this paper, we need to introduce them only for increasing properties. For further details see [10]. Let Q be an increasing property of graphs from 2^{K_n}. Let $p = p(n)$ be a function. A function $\widehat{p} = \widehat{p}(n)$ is called a *threshold function* for Q iff the following condition holds:

$$\Pr[\, \mathbb{G}(n,p) \text{ has } Q \,] \to \begin{cases} 0 & \text{if} \quad p \ll \widehat{p}\,, \\ 1 & \text{if} \quad p \gg \widehat{p}\,. \end{cases}$$

Threshold functions play a crucial role in examination of the phase transition phenomena in random graphs [3, 10, 11, 15]. A typical property is the connectivity of random graphs and more specifically, size of connected components. The threshold function for the existence of a giant component is $\widehat{p} = 1/n$. As stated in [15], Sect. 4, a random graph contains a.a.s. the largest component with $\Theta(n^{2/3})$ vertices in its *critical phase*, i.e. if $p = 1/n$. The *subcritical phase* is for $p = c/n$ where $0 < c < 1$. It represents such a state that all components are trees or unicyclic and the largest component is of size $\Theta(\log n)$ a.a.s. Otherwise, if $p = c/n$ where $c > 1$, then there is a unique giant component of order $\Theta(n)$ a.a.s. in the *supercritical phase*. The rest of the random graph consists of "small" trees and as c increases, the giant component grows by absorbing the trees.

3 Graphlets

Let X_{g_i} be the random variable on $\mathbb{G}(n,p)$ denoting the number of copies of graphlet g_i in a random graph for $i = 0, \ldots, 9$. In this paper, we consider 10 graphlets with at most 4 vertices (see Fig. 1) that is why $i = 0, \ldots, 9$.

Lemma 1 ([10]). *For $i = 0, \ldots, 9$, let n_i (m_i) denote the number of vertices (edges) of g_i. The expectation of the random variable X_{g_i} is given by*

$$\mathbb{E}(X_{g_i}) = \frac{n_i!}{aut(g_i)} \binom{n}{n_i} p^{m_i} (1-p)^{\binom{n_i}{2} - m_i}\,, \qquad (2)$$

where $aut(g_i)$ denotes the number of automorphisms of g_i.

Vertices (i.e. graphlets g_0) represent a trivial case, thus we will consider only random variables X_{g_i} for $i > 0$ in this section. We shall examine the phase transition behavior of graphlets at first. The occurrence of a given graphlet is not a monotone property [10]. Nevertheless, such a property has two threshold functions, one in sparse random graphs and the second in very dense random graphs [10]. We are interested only in the first case since almost all real networks are sparse.

For a given graph G, let $\tau(G)$ denote the ratio of the number of edges to the number of vertices in the densest subgraph of G, i.e.

$$\tau(G) = \max \left\{ \frac{|E(H)|}{|V(H)|}\,;\, H \subseteq G, |V(H)| > 0 \right\}.$$

The following statement determines threshold functions for graphlets.

Theorem 1 ([10]). *For an arbitrary graphlet g with at least one edge, it holds*

$$\lim_{n\to\infty} \Pr[\, g \textit{ occurs in } \mathbb{G}(n,p) \,] = \begin{cases} 0 & \textit{if} \quad p \ll n^{-1/\tau(g)}\,, \\ 1 & \textit{if} \quad p \gg n^{-1/\tau(g)}\,. \end{cases}$$

As a consequence, we obtain Tab. 1 in which the thresholds functions for all nontrivial graphlets are listed. Determination of threshold functions is straightforward because none of graphlets (for $i = 1, \ldots, 9$) contains a denser subgraph than itself. Clearly, $\tau(g_1) = 1/2$ and $\tau(g_2) = \tau(P_3) = 2/3$. By the same argument, $\tau(g_3) = \tau(\triangle) = 3/3 = 1$, etc.

Table 1: Threshold functions of nontrivial graphlets.

Graphlet g_i	Description	Threshold function
g_1	Edge	n^{-2}
g_2	Path P_3	$n^{-3/2}$
g_3	Triangle $\triangle$	n^{-1}
g_4	Path P_4	$n^{-4/3}$
g_5	3-star	$n^{-4/3}$
g_6	Cycle C_4	n^{-1}
g_7	$\triangle$+edge	n^{-1}
g_8	Chordal-C_4	$n^{-4/5}$
g_9	Clique K_4	$n^{-2/3}$

If we consider sparse random graphs with $p = c/n$, then it is possible to deduce from Tab. 1 which graphlets occur more or less frequently. We can see that the presence of trees (i.e. graphlets g_1, g_2, g_4 and g_5) is the most probable of all graphlets in $\mathbb{G}(n, c/n)$. On the other hand, dense graphlets (such as the clique g_9) rarely occur in $\mathbb{G}(n, c/n)$.

The expected number (or "average counts") of graphlets can be derived from Lemma 1. The probabilities of graphlets and corresponding expectations of the random variable X_{g_i} are listed in Tab. 2. The numbers of automorphisms $aut(g_i)$ for these small graphs are well-known (see [19] for the details).

Table 2: Graphlets, numbers of their automorphisms, probabilities of graphlets and values of $\mathbb{E}(X_{g_i})$.

Graphlet g_i	$aut(g_i)$	Probability of g_i	$\mathbb{E}(X_{g_i})$
g_1	2	p	$\binom{n}{2}p$
g_2	2	$p^2(1-p)$	$\frac{3!}{2}\binom{n}{3}p^2(1-p)$
g_3	6	p^3	$\frac{3!}{6}\binom{n}{3}p^3$
g_4	2	$p^3(1-p)^3$	$\frac{4!}{2}\binom{n}{4}p^3(1-p)^3$
g_5	6	$p^3(1-p)^3$	$\frac{4!}{6}\binom{n}{4}p^3(1-p)^3$
g_6	8	$p^4(1-p)^2$	$\frac{4!}{8}\binom{n}{4}p^4(1-p)^2$
g_7	2	$p^4(1-p)^2$	$\frac{4!}{2}\binom{n}{4}p^4(1-p)^2$
g_8	4	$p^5(1-p)$	$\frac{4!}{4}\binom{n}{4}p^5(1-p)$
g_9	24	p^6	$\frac{4!}{24}\binom{n}{4}p^6$

4 Isolated Graphlets

Given a graph $G=(V,E)$, a graphlet is said to be an *isolated graphlet* if it is a component in G. Let Y_{g_i} be the random variable on $\mathbb{G}(n,p)$ denoting the number of copies of isolated graphlet g_i in a random graph for $i=0,\ldots,9$. It will be seen later (in Tab. 3) that it is meaningful to take into account the isolated graphlet g_0 as well.

Lemma 2 ([10])**.** *For $i=0,\ldots,9$, let n_i (m_i) denote the number of vertices (edges) of g_i. The expectation of the random variable Y_{g_i} is given by*

$$\mathbb{E}(Y_{g_i}) = \mathbb{E}(X_{g_i})\cdot(1-p)^{(n-n_i)n_i}\,. \tag{3}$$

In order to express an asymptotic estimation for "average counts" of isolated graphlets in random graphs, the following lemma is necessary.

Lemma 3. *Let α,β,c be constants (i.e. $\alpha,\beta,c \ll n$) such that $\alpha,c>0$ and let $p=c/n$. It holds*

$$(1-p)^{\alpha n+\beta} \sim e^{-\alpha c} \qquad as \quad n\to\infty\,.$$

Proof. By assumptions of Lemma,

$$(1-p)^{\alpha n+\beta} = \left(1-\frac{c}{n}\right)^{\alpha n}\cdot\left(1-\frac{c}{n}\right)^{\beta}\,.$$

Thus

$$(1-p)^{\alpha n+\beta} = \left(1+\frac{1}{-\frac{n}{c}}\right)^{\frac{-\alpha nc}{-c}}\cdot\left(1-\frac{c}{n}\right)^{\beta} \sim e^{-\alpha c}$$

since

$$\lim_{n\to\infty}\left(1+\frac{1}{-\frac{n}{c}}\right)^{-\frac{n}{c}} = e$$

and

$$\lim_{n\to\infty}\left(1-\frac{c}{n}\right)^{\beta} = 1\,.$$

□

By Lemma 2 and 3, we derive the following statement. It determines the asymptotic estimation for expected number of isolated graphlets in random graphs with $p=c/n$.

Lemma 4. *Let $c>1$ be constant. There exists a function $\psi_c(n)=O(n^{-1})$ such that for each g_i (with n_i vertices and m_i edges) in $\mathbb{G}(n,c/n)$ it holds*

$$\mathbb{E}(Y_{g_i}) \sim \begin{cases} nc^{n_i-1}e^{-cn_i}/aut(g_i) & \text{if} \quad m_i = n_i-1, \\ c^{n_i}e^{-cn_i}/aut(g_i) & \text{if} \quad m_i = n_i, \\ \psi_c(n) & \text{if} \quad m_i \geq n_i+1. \end{cases}$$

This lemma allows for asymptotic estimation of expected numbers of isolated graphlets in random graphs $\mathbb{G}(n,c/n)$. Corresponding estimations are listed in Tab. 3. One may observe that the frequency of isolated trees (i.e. graphlets g_0,g_1,g_2,g_4,g_5) growth linearly with n, the frequency of isolated graphlets g_3,g_6,g_7 (i.e. trees with one additional edge) is constant with respect to n and the frequency of other isolated graphlets (g_8 and g_9) is negligible. The intuition behind this result is, similarly as in the previous section, that the contribution of sparse isolated graphlets is more significant (even in magnitude) than of denser ones. Unless as in the previous section, the frequency distribution of isolated vertices g_0 can be expressed by the asymptotic formula which depends on n and c. (Note that the count of graphlets g_0 is trivially n.)

Table 3: Asymptotic estimations of expected number for isolated graphlets in $\mathbb{G}(n,c/n)$ as n is large enough.

Graphlet g_i	$\mathbb{E}(Y_{g_i})$	Estimation of $\mathbb{E}(Y_{g_i})$ for $p=c/n$
g_0	$n(1-p)^{n-1}$	ne^{-c}
g_1	$\binom{n}{2}p(1-p)^{2(n-2)}$	$\frac{1}{2}nce^{-2c}$
g_2	$\frac{3!}{2}\binom{n}{3}p^2(1-p)^{3(n-3)+1}$	$\frac{1}{2}nc^2e^{-3c}$
g_3	$\frac{3!}{6}\binom{n}{3}p^3(1-p)^{3(n-3)}$	$\frac{1}{6}c^3e^{-3c}$
g_4	$\frac{4!}{2}\binom{n}{4}p^3(1-p)^{4(n-4)+3}$	$\frac{1}{2}nc^3e^{-4c}$
g_5	$\frac{4!}{6}\binom{n}{4}p^3(1-p)^{4(n-4)+3}$	$\frac{1}{6}nc^3e^{-4c}$
g_6	$\frac{4!}{8}\binom{n}{4}p^4(1-p)^{4(n-4)+2}$	$\frac{1}{8}c^4e^{-4c}$
g_7	$\frac{4!}{2}\binom{n}{4}p^4(1-p)^{4(n-4)+2}$	$\frac{1}{2}c^4e^{-4c}$
g_8	$\frac{4!}{4}\binom{n}{4}p^5(1-p)^{4(n-4)+1}$	$\frac{1}{4}n^{-1}c^5e^{-4c}$
g_9	$\frac{4!}{24}\binom{n}{4}p^6(1-p)^{4(n-4)}$	$\frac{1}{24}n^{-2}c^6e^{-4c}$

5 Domination Number in Sparse Random Graphs

A *dominating set* of a graph $G=(V,E)$ is a set $D\subseteq V(G)$ such that every vertex not in D is adjacent to at least one vertex of D. The *domination number*, denoted by $\gamma(G)$, is the minimum cardinality of a dominating set of G.

Results regarding domination problems and domination number are surveyed in [7]. Due to significant applications in social, engineering and PPI networks, the area of domination became recently attractive and fast growing [5, 12, 13, 14, 18].

It was shown in [18] that the domination number of random graphs *for a constant* p may attain one of only two possible values. Such a property is called the *two-point concentration*. The two-point concentration result was recently extended for random graphs $\mathbb{G}(n,p)$ with $p \gg \frac{\ln^2 n}{\sqrt{n}}$ (in this case, $p = p(n)$ is assumed to be a function). However, it does not hold if $p = O(\log n/n)$ [5]. As mentioned in [5], the detailed analysis of the domination number behavior for random graphs with $p \ll \frac{1}{\sqrt{n}}$ is still an interesting open problem.

In order to pursuit this problem, we suggest to use a method based on isolated graphlet counting. Recall that a random graph consists of a single giant component and small isolated trees a.a.s. in its critical and supercritical phase if $p = \Theta(n^{-1})$. Roughly speaking, our idea resides in exact counting of domination numbers for isolated trees and its estimation for the giant component. Resulting bounds could be obtained by a combination of all particular estimations. Our preliminary results exploiting the early version of this idea have been published in [14].

6 Concluding Remarks

One possible extension of this work may involve analysis for graphlets with 5 vertices. Although the idea is the same as for smaller graphlets, detailed calculations need an additional effort because there are 21 graphlets with 5 vertices.

Comparing actual networks to random graphs might be a meaningful goal for future research. A resulting knowledge might be helpful to better understanding of real-world networks structure.

As regards the problem of the domination number estimation, the author is currently working on more accurate formulas of results published in [14]. The corresponding analysis is based on the idea which was explained in the previous section.

Acknowledgement. The support from the Slovak Scientific Grant Agency under the grant VEGA 1/0026/16 is gratefully acknowledged.
The author thanks the anonymous referees for their valuable comments on the manuscript.

References

[1] B. Bollobás: Random Graphs. (2^{nd} edition) Cambridge Studies in Advanced Mathematics 73, (2001)

[2] J. Dreier, P. Kuinke, B. Le Xuan, P. Rossmanith: Local Structure Theorems for Erdős-Rényi Graphs and Their Algorithmic Applications. In Proc. of the 44^{th} Int. Conference on Current Trends in Theory and Practice of Comp. Science, SOFSEM 2018, LNCS 10706, Springer Int. Publishing, 2018, 125–136

[3] P. Erdős, A. Rényi: On the evolution of random graphs. Publ. Math. Inst. Hungar. Acad. Sci. **5** (1960) 17–61

[4] M. R. Garey, D. S. Johnson: Computers and Intractability. Freeman, New York (1979)

[5] R. Glebov, A. Liebenau, T. Szabó: On the Concentration of the Domination Number of the Random Graph. SIAM J. Discrete Math., **29**:3 (2015) 1186–1206

[6] J.L. Gross, J. Yellen: Handbook of Graph Theory. CRC Press (2003)

[7] T. W. Haynes, S. T. Hedetniemi, P. J. Slater: Fundamentals of Domination in Graphs. Marcel Dekker, Inc., New York (1998)

[8] T. Hočevar: Graphlet Counting. In Proc. of the 17^{th} Conference on Applied Mathematics, APLIMAT 2018, SUT in Bratislava, 2018, 442–449

[9] T. Hočevar, J. Demšar: Computation of Graphlet Orbits for Nodes and Edges in Sparse Graphs. Journal of Statistical Software **71**:10 (2016)

[10] S. Janson, T. Łuczak, A. Rucinski: Random Graphs. John Wiley & Sons, New York (2000)

[11] M. Karoński: Random Graphs. Handbook of Combinatorics (R.L. Graham, M. Grötschel, L. Lovász eds.), Vol I, Elsevier, 1995, 351–380

[12] F. Molnár, S. Sreenivasan, B. K. Szymanski, G. Korniss: Minimum Dominating Sets in Scale-Free Network Ensembles. Scientific Reports 3, Article number: 1736 (2013)

[13] J. C. Nacher, T. Akutsu: Dominating scale-free networks with variable scaling exponent: heterogeneous networks are not difficult to control. New Journal of Physics 14, 073005 (2012)

[14] M. Nehéz, D. Bernát, M. Lelovský: Estimation of the domination number in sparse random graphs and applications. In Proc. of the 5^{th} European Conference on the Engineering of Computer-Based Systems, ECBS 2017, ACM New York, 2017, 7:1–7:9

[15] E. M. Palmer: Graphical Evolution. John Wiley & Sons, Inc., New York (1985)

[16] N. Przulj, D.G. Corneil, I. Jurisica: Modeling interactome: scale-free or geometric?. Bioinformatics **20**:18 (2004) 3508–3515

[17] A. Rucinski: Small subgraphs of random graphs. Presentation of the talk, Joint Mathematics Meeting #1003, MAA 2005, Atlanta (2005) URL: `https://www.math.cmu.edu/ãf1p/MAA2005/L4.pdf`

[18] B. Wieland, A.P. Godbole: On the Domination Number of a Random Graph. Electronic Journal of Combinatorics **8**:1 (2001) #R37

[19] Wolfram MathWorld: Graph Automorphism. URL: `http://mathworld.wolfram.com/GraphAutomorphism.html` (2018)

S. Krajči (ed.): ITAT 2018 Proceedings, pp. 176–180
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Robert Jajcay, Tatiana Jajcayová, and Marián Opial

Enumeration of matrices with prohibited bounded sub-windows

Robert Jajcay, Tatiana Jajcayová, and Marián Opial

Faculty of Mathematics, Physics and Informatics, Comenius University, Bratislava, Slovakia
robert.jajcay@fmph.uniba.sk, jajcayova@fmph.uniba.sk, opialm@gmail.com

Abstract: Let $\mathscr{A}$ be a finite alphabet, and let $\mathscr{S}$ be a set of of 2-dimensional bounded prohibited patterns over $\mathscr{A}$. We consider the set $\mathscr{M}_{\mathscr{A},\mathscr{S}}$ of matrices over $\mathscr{A}$ that avoid the patterns from $\mathscr{S}$, and attempt to derive (closed or linear recurrence) formulas for the numbers of $m \times n$ matrices in $\mathscr{M}_{\mathscr{A},\mathscr{S}}$. We argue that different sets of prohibited patterns require different types of formulas, with some formulas recurrent in just one of the parameters m,n, some satisfying a two-dimensional linear recurrence relation (depending of both m and n), and some satisfying neither of the two types. We consider characterization of classes that admit a two-dimensional linear recurrence relation, as well as classes that do not allow for such relation. In addition, given $\mathscr{A}$ and $\mathscr{S}$, we address the question of the existence of a constant a such that the number of $m \times n$ matrices in $\mathscr{M}_{\mathscr{A},\mathscr{S}}$ is asymptotically equal to $|\mathscr{A}|^{amn}$.

We report on preliminary results for a specific class of boolean matrices with the prohibited set consisting of thirty-two 3×3 matrices for which computational results suggest the non-existence of a two-dimensional linear recurrence relation.

1 Introduction and preliminaries

Many classes of objects are defined via prohibiting specified sub-objects. In our paper, we deal with classes of matrices over finite alphabets that do not contain patterns from a finite set of local prohibited patterns. Such matrices can be viewed as matrices recognizable via a bounded window automaton with a finite memory that can only view a bounded area of the matrix at a time and cannot see (or remember) the matrix in its entirety (while it is allowed to slide through the entire matrix window by window, verifying each window separately). The motivation behind considering these classes of matrices lies in extending the theory of 'one-dimensional' languages of strings avoiding specified substrings to two dimensional arrays. One-dimensional languages that avoid (connected) substrings from a finite set of prohibited substrings have been studied for several decades and their enumeration is well-known to lead to homogeneous linear recurrence relations (see. e.g., [3, 4]), We show that a similar, although more complicated, situation holds in the case of two-dimensional arrays. We stress that when talking of submatrices, we mean connected blocks.

Let $\mathscr{A}$ be a finite alphabet, and let $\mathscr{S}$ be a set of $k \times \ell$ matrices over $\mathscr{A}$, $k,\ell \geq 1$. Let $\mathscr{M}_{\mathscr{A},\mathscr{S}}$ denote the set matrices over $\mathscr{A}$ that do not contain (avoid) sub-matrices from $\mathscr{S}$, i.e., the set of matrices $\mathbf{A} = \| a_{i,j} \|_{m.n}$, $a_{i,j} \in \mathscr{A}$, for $1 \leq i \leq m$, $1 \leq j \leq m$, having the property that none of the $k \times \ell$ submatrices of $\mathbf{A}$ belong to $\mathscr{S}$ (thus, k and ℓ are the dimensions of the viewing window of the automaton recognizing $\mathbf{A}$; it accepts $\mathbf{A}$ if and only if it never finds a matrix from $\mathscr{S}$ in its viewing window).

We illustrate this concept with a specific class of matrices with prohibited patterns that will be used throughout our paper.

Example 1. *Let $\mathscr{A} = \{0,1\}$, and consider the set of boolean matrices over $\mathscr{A}$ not admitting 3×3 crosses of zeroes or ones, i.e., not admitting submatrices of the form*

$*$	0	$*$
0	0	0
$*$	0	$*$

$*$	1	$*$
1	1	1
$*$	1	$*$

where the stars stand for arbitrary elements from $\mathscr{A}$ (to avoid using stars, one could think of the set of the 32 prohibited matrices obtained by making all the possible choices). We will call the matrices from this class noise matrices*, and note that they are often considered to be examples of chaotic, structure-less matrices.*

Given an alphabet $\mathscr{A}$ and a set $\mathscr{S}$ of prohibited $k \times \ell$ submatrices over $\mathscr{A}$, let $N_{\mathscr{A},\mathscr{S}}(m,n)$ denote the number of $m \times n$ matrices in $\mathscr{M}_{\mathscr{A},\mathscr{S}}$. Then clearly $N_{\mathscr{A},\mathscr{S}}(m,n) = |\mathscr{A}|^{mn}$, for all $1 \leq m \leq k$ and $1 \leq n \leq \ell$, with at least one parameter smaller than the upper bound, while

$$0 \leq N_{\mathscr{A},\mathscr{S}}(m,n) \leq |\mathscr{A}|^{mn} \qquad (1)$$

in general.

In what follows, we are interested in deriving formulas for $N_{\mathscr{A},\mathscr{S}}(m,n)$ for various alphabets $\mathscr{A}$ and sets of prohibited sub-matrices $\mathscr{S}$.

Example 2. *Considering $\mathscr{A}=\{0,1\}$ again, taking empty $\mathscr{S}_1$ yields $\mathscr{M}_{\mathscr{A},\mathscr{S}_1}$ consisting of all boolean matrices and $N_{\mathscr{A},\mathscr{S}_1}(m,n)=2^{mn}$, for all m and n.*

Taking $\mathscr{S}_2$ to consist of the single 1×1 matrix with $a_{1,1}=1$ implies that $\mathscr{M}_{\mathscr{A},\mathscr{S}_2}$ consists of just the $m\times n$ zero-matrices and $N_{\mathscr{A},\mathscr{S}_2}(m,n)=1$, for all m and n.

Finally, taking $\mathscr{S}_3$ to consist of the 2×2 all-ones matrix $a_{i,j}=1$, for $1\le i,j\le 2$, yields $\mathscr{M}_{\mathscr{A},\mathscr{S}_3}$ consisting of all 1×1, 1×2, 2×1 matrices, and $m\times n$ matrices that do not contain a 2×2 sub-matrix of all ones, for $m,n\ge 2$. Thus,

$$N_{\mathscr{A},\mathscr{S}_3}(1,1)=2,$$

$$N_{\mathscr{A},\mathscr{S}_3}(1,2)=N_{\mathscr{A},\mathscr{S}_3}(2,1)=2^2=4,$$

$$N_{\mathscr{A},\mathscr{S}_3}(2,2)=2^4-1=15,$$

and the Inclusion-Exclusion Principle yields that $N_{\mathscr{A},\mathscr{S}_3}(2,3)=2^6-2^2-2^2+1=57$.

One of the main conjectures concerning the asymptotic behavior of the numbers $N_{\mathscr{A},\mathscr{S}}(m,n)$ states the following:

Conjecture 1. *Let $\mathscr{A}$ be a finite alphabet, and let $\mathscr{S}$ be a set of prohibited $k\times\ell$ submatrices over $\mathscr{A}$. Then there exists a constant $0\le a\le 1$ such that*

$$\lim_{m\to\infty,n\to\infty}\frac{N_{\mathscr{A},\mathscr{S}}(m,n)}{|\mathscr{A}|^{amn}}=1.$$

If the a from the above conjecture exists for a specific pair $\mathscr{A}$ and $\mathscr{S}$, we say that a is the *critical exponent* for the pair. The sets $\mathscr{S}_1$ and $\mathscr{S}_2$ defined in Example 2 constitute extremal cases with the critical exponents $a_1=1$ and $a_2=0$, respectively.

The paper [1] contains the following information about the asymptotic behavior of the enumeration function of the noise matrices.

Theorem 1 ([1])**.** *Let $\mathscr{A}$ and $\mathscr{S}$ be those defined in Example 1. For every $m\ge 3$, there exists a constant $0\le a_m\le 1$ such that*

$$\lim_{n\to\infty}\frac{N_{\mathscr{A},\mathscr{S}}(m,n)}{|\mathscr{A}|^{a_m mn}}=1.$$

Furthermore, there exist two constants $0<b_1<b_2<1$ such that

$$2^{b_1 mn}<N_{\mathscr{A},\mathscr{S}}(m,n)<2^{b_2 mn},$$

for all $m,n\ge 3$.

While computer experimentation appears to support Conjecture 1, in principle, it cannot be used to prove the claim for any specific pair $\mathscr{A}$ and $\mathscr{S}$. However, it usually fairly quickly provides for estimates for the value of a. In particular, finding the numbers $N_{\mathscr{A},\mathscr{S}}(m,n)$ for a large range of pairs m,n allows one to calculate $\frac{\log_{|\mathscr{A}|}(N_{\mathscr{A},\mathscr{S}}(m,n))}{mn}$ for each such pair. The actual values for large pairs often match for a considerable number of decimal places. For example, calculations concerning the enumeration of noise matrices reported in [6] yield that the corresponding a (if it exists!) lies in the range:

$$0.9068\le a\le 0.947564.$$

2 One-dimensional linear recurrence relations

Let $\mathscr{A}$ and $\mathscr{S}$ be a finite alphabet and a set of $k\times\ell$ prohibited matrices over $\mathscr{A}$. In this section, we prove that for any given $m\ge k$ there exists a linear recurrence formula tying together the numbers $N_{\mathscr{A},\mathscr{S}}(m,n)$, $n\ge 1$. We use a generalization of a technique used in [1] for noise matrices.

Example 3. *To illustrate the basic idea of this approach, suppose we extend a $3\times n$ matrix ending in a specific triple of columns by adding a specific new column which results in a $3\times(n+1)$ matrix ending in a new triple of columns (but sharing two columns with the original triple). We may encounter two different situations:*

$$\cdots\begin{array}{|c|c|c|}\hline 0&1&1\\\hline 1&1&1\\\hline 0&0&1\\\hline\end{array}+\begin{array}{|c|}\hline 0\\\hline 0\\\hline 1\\\hline\end{array}=\cdots\begin{array}{|c|c|c|}\hline 1&1&0\\\hline 1&1&0\\\hline 0&1&1\\\hline\end{array}$$

or

$$\cdots\begin{array}{|c|c|c|}\hline 0&1&1\\\hline 1&1&1\\\hline 0&0&1\\\hline\end{array}+\begin{array}{|c|}\hline 0\\\hline 1\\\hline 1\\\hline\end{array}=\cdots\begin{array}{|c|c|c|}\hline 1&1&0\\\hline 1&1&1\\\hline 0&1&1\\\hline\end{array}$$

When considering the noise matrices defined in Example 1, these two situations differ as follows. Any noise matrix ending in the first triple remains

a noise matrix after adding the specified column, while the matrix formed from a noise matrix by adding the second specified column ceases being a noise matrix.

With regard to the above example, it is important to point out that the entire situation only depends of the last three columns, and the actual number of columns of the matrices is irrelevant with regard to the above claims.

In view of these observations, let $\mathscr{A}$ be a finite alphabet and $\mathscr{S}$ be a set of $k \times \ell$ prohibited matrices over $\mathscr{A}$, and let us fix the number $m \geq k$ of rows. Let $\{\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_{|\mathscr{A}|^{m\ell}}\}$ be the set of all $m \times \ell$ matrices over $\mathscr{A}$ (listed in an arbitrary but fixed order). For each $n \geq \ell$, divide the $m \times n$ matrices in $\mathscr{M}_{\mathscr{A},\mathscr{S}}$ with regard to their last ℓ columns, and let α_i^n denote the number of $m \times n$ matrices in $\mathscr{M}_{\mathscr{A},\mathscr{S}}$ ending in the matrix $\mathbf{M}_i$, $1 \leq i \leq |\mathscr{A}|^{m\ell}$. Denote $\alpha(n) = (\alpha_1^n, \alpha_2^n, \ldots, \alpha_{|\mathscr{A}|^{m\ell}}^n)$, and note that

$$\alpha(n) \cdot \mathbf{1}^T = \sum_{i=1}^{|\mathscr{A}|^{m\ell}} \alpha_i^n = N_{\mathscr{A},\mathscr{S}}(m,n), \qquad (2)$$

(where $\mathbf{1}^T$ stands for the column of all ones).

As observed above, if we expand an $m \times n$ matrix ending in $\mathbf{M}_i$ by adding a column, we obtain an $m \times (n+1)$ matrix ending in $\mathbf{M}_j$ having the property that the first $\ell - 1$ columns of $\mathbf{M}_j$ match the last $\ell - 1$ columns of $\mathbf{M}_i$. If this is the case, we will say that $\mathbf{M}_j$ is a *successor* of $\mathbf{M}_i$. Moreover, if $n \geq \ell$, the question whether an $m \times n$ matrix in $\mathscr{M}_{\mathscr{A},\mathscr{S}}$ ending in $\mathbf{M}_i$ remains in $\mathscr{M}_{\mathscr{A},\mathscr{S}}$ after a column is added to it so that it ends in $\mathbf{M}_j$ depends of $\mathbf{M}_i$ and $\mathbf{M}_j$ only and it is *independent* of the number of columns n. Therefore, for $1 \leq i, j \leq |\mathscr{A}|^{m\ell}$, let $a_{i,j} = 1$ if $\mathbf{M}_j$ is a successor of $\mathbf{M}_i$ having the property that if an $m \times n$ matrix ending in $\mathbf{M}_i$ belongs to $\mathscr{M}_{\mathscr{A},\mathscr{S}}$ then so does the $m \times (n+1)$ matrix ending in $\mathbf{M}_j$ (constructed from the smaller matrix by adding a column). Let $a_{i,j} = 0$ otherwise, and denote $\mathbb{A} = \| a_{i,j} \|$. Since every $m \times (n+1)$ matrix in $\mathscr{M}_{\mathscr{A},\mathscr{S}}$ is obtained from a specific $m \times n$ matrix in $\mathscr{M}_{\mathscr{A},\mathscr{S}}$, it follows that

$$\alpha(n+1) = \alpha(n)\mathbb{A}, \qquad (3)$$

for all $n \geq \ell$.

Suppose now that the square matrix $\mathbb{A}$ is a root of a monic polynomial $p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{s-1}x^{s-1} + x^s$ in $\mathbb{Z}[x]$, i.e.,

$$a_0\mathbb{I} + a_1\mathbb{A} + a_2\mathbb{A}^2 + \ldots + a_{s-1}\mathbb{A}^{s-1} + \mathbb{A}^s = \mathbb{O},$$

where $\mathbb{I}$ stands for the identity matrix and $\mathbb{O}$ for the all-zeroes matrix. Thus,

$$\mathbb{A}^s = -a_0\mathbb{I} - a_1\mathbb{A} - a_2\mathbb{A}^2 - \ldots - a_{s-1}\mathbb{A}^{s-1}, \quad (4)$$

and after multiplying by $\alpha(n)$ on the left and by $\mathbf{1}^T$ on the right we obtain

$$\alpha(n)\mathbb{A}^s\mathbf{1}^T = -a_0\alpha(n)\mathbf{1}^T - a_1\alpha(n)\mathbb{A}\mathbf{1}^T - a_2\alpha(n)\mathbb{A}^2\mathbf{1}^T - \ldots - a_{s-1}\alpha(n)\mathbb{A}^{s-1}\mathbf{1}^T.$$

Applying equations (2) and (3) finally yields

$$N_{\mathscr{A},\mathscr{S}}(m,n+s) = -a_0 N_{\mathscr{A},\mathscr{S}}(m,n) - \ldots - a_{s-1}N_{\mathscr{A},\mathscr{S}}(m,n+s-1), \qquad (5)$$

which is a linear recurrence relation.

The above arguments allow us to prove the following generalization of Theorem 2 to *all* sets of matrices over finite alphabets with prohibited bounded patterns.

Theorem 2 ([1])**.** *Let $\mathscr{A}$ be a finite alphabet and $\mathscr{S}$ be a set of $k \times \ell$ prohibited matrices over $\mathscr{A}$. For every $m \geq k$, there exists a linear recurrence relation such that*

$$N_{\mathscr{A},\mathscr{S}}(m,n+s) = -a_0 N_{\mathscr{A},\mathscr{S}}(m,n) - \ldots - a_{s-1}N_{\mathscr{A},\mathscr{S}}(m,n+s-1),$$

as well as a constant $0 \leq c_m \leq 1$ such that

$$\lim_{n\to\infty} \frac{N_{\mathscr{A},\mathscr{S}}(m,n)}{|\mathscr{A}|^{c_m mn}} = 1.$$

Proof. Let $\mathscr{A}$ and $\mathscr{S}$ be as stated, and suppose that $m \geq k$. The matrix $\mathbb{A}$ defined in the discussion preceding the statement of our theorem is an $|\mathscr{A}|^{m\ell} \times |\mathscr{A}|^{m\ell}$ boolean matrix which (by the Cayley-Hamilton theorem) is the root of its characteristic polynomial $char_{\mathbb{A}}(x)$, which belongs to $\mathbb{Z}[x]$, and is either monic when $|\mathscr{A}|^{m\ell}$ is even or can be made monic by multiplying by -1 when $|\mathscr{A}|^{m\ell}$ is odd. This yields a linear recurrence relation of order $|\mathscr{A}|^{m\ell}$ for the numbers $N_{\mathscr{A},\mathscr{S}}(m,n)$, $n \geq \ell$. Since $\mathbb{A}$ is a boolean (i.e., non-negative) matrix, using the Perron-Frobenius theorem yields that its spectral radius $\rho(\mathbb{A})$ is its eigenvalue of the largest modulus. Consequently, $\rho(\mathbb{A})$ determines the magnitude of any sequence satisfying the recurrence relation determined by $char_{\mathbb{A}}(x)$ [2, 5], therefore $N_{\mathscr{A},\mathscr{S}}(m,n) = \theta(\rho(\mathbb{A})^n)$, and the second claim of our theorem follows. □

3 One-dimensional linear recurrence relations of smallest order

Even though we have proved the existence of an one-dimensional linear recurrence relation for each $\mathscr{A},\mathscr{S}$, and $m \geq k$, the orders $|\mathscr{A}|^{m\ell}$ of these relations are rather large. The key problem when using such relations lies in the need to find the first $|\mathscr{A}|^{m\ell}$ elements of the corresponding sequence by brute force. Thus, in order to start using the above described recurrence relation for the numbers $N_{\mathscr{A},\mathscr{S}}(3,n)$ in the case of noise matrices (for which the prohibited matrices are of dimension 3×3), one first needs to find the numbers

$$N_{\mathscr{A},\mathscr{S}}(3,3), N_{\mathscr{A},\mathscr{S}}(3,4), \ldots, N_{\mathscr{A},\mathscr{S}}(3,2^9),$$

which turns out to be a computationally demanding task simply because of the sheer size of the search spaces and the corresponding frequency numbers. For example, $N_{\mathscr{A},\mathscr{S}}(3,55000) \approx 2^{0.970956\cdot 3\cdot 55000} \approx 2^{160207}$, and 640 MB of memory space were needed to store the first $55,000$ members of the sequence $N_{\mathscr{A},\mathscr{S}}(3,n)$ [6]. (Clearly, in order to obtain the correct recurrence relation, one needs to calculate and store the exact numbers.)

Finding recurrence relations of smaller degrees is therefore of utter importance. The first obvious choice for reducing the degree of the obtained recurrence relation is to use the minimal polynomial for $\mathbb{A}$ over $\mathbb{C}$ instead of its characteristic polynomial. However, while calculating the characteristic polynomial for $\mathbb{A}$ is a computationally demanding but simple determinant calculation, finding the minimal polynomial for $\mathbb{A}$ requires finding the roots for $char_{\mathbb{A}}(x)$ or its irreducible divisors. Moreover, the minimal polynomial over $\mathbb{C}$ most likely does not belong to $\mathbb{Z}[x]$, making the exact calculation of the coefficients of the corresponding recurrence relation impossible. While this problem can be remedied by considering the minimal polynomial over $\mathbb{Q}$ (which does belong to $\mathbb{Z}[x]$), in general, this would be of higher degree than the minimal polynomial over $\mathbb{C}$, and still hard to find.

In [6], the third author under the supervision of the second author of this article considered the noise matrices and chose a much simpler computational approach. Using essentially brute force, he found the numbers of noise matrices $N_{\mathscr{A},\mathscr{S}}(3,n)$ for $1 \leq n \leq 55000$. Having the numbers from this list, he created a list consisting of the numbers $\frac{\log_2(N(3,n))}{3n}$, looking for a pattern. An easy inspection reveals that $\frac{\log_2(N(3,1500))}{3\cdot 1500} \approx 0.970992$, while $\frac{\log_2(N(3,55000))}{3\cdot 55000} \approx 0.970956$; the critical exponent for $m=3$ becomes exact up to the first four decimal digits fairly quickly.

Similarly, calculating the numbers $N_{\mathscr{A},\mathscr{S}}(4,n)$ for $1 \leq n \leq 35000$ determined the critical exponent for $m=4$ equal to 0.959452; the numbers $N_{\mathscr{A},\mathscr{S}}(5,n)$ for $1 \leq n \leq 50000$ determined the critical exponent for $m=5$ equal to 0.952307; and finally calculating the numbers $N_{\mathscr{A},\mathscr{S}}(6,n)$ for $1 \leq n \leq 25000$ determined the critical exponent for $m=6$ equal to 0.9475645.

As for the recurrence relation of minimal degree, having the actual values of the corresponding sequence allows one to find the minimal degree experimentally. Specifically, let $k \geq 2$, and suppose an equivalence relation of degree k exists. If that were the case, the solution $a_0, a_1, a_2, \ldots, a_{k-1}$ of the $k\times k$ system of linear equations

$$\begin{aligned}
a_0 N_{\mathscr{A},\mathscr{S}}(m,\ell) + \ldots + a_{k-1} N_{\mathscr{A},\mathscr{S}}(m,\ell+k-1) &= N_{\mathscr{A},\mathscr{S}}(m,\ell+k)\\
a_0 N_{\mathscr{A},\mathscr{S}}(m,\ell+1) + \ldots + a_{k-1} N_{\mathscr{A},\mathscr{S}}(m,\ell+k) &= N_{\mathscr{A},\mathscr{S}}(m,\ell+k+1)\\
&\ldots\\
a_0 N_{\mathscr{A},\mathscr{S}}(m,\ell+k-1) + \ldots + a_{k-1} N_{\mathscr{A},\mathscr{S}}(m,\ell+2k-2) &= N_{\mathscr{A},\mathscr{S}}(m,\ell+2k-1)
\end{aligned}$$

would have to satisfy all the 'latter' systems, $i \geq 1$,

$$\begin{aligned}
a_0 N_{\mathscr{A},\mathscr{S}}(m,\ell+i) + \ldots + a_{k-1} N_{\mathscr{A},\mathscr{S}}(m,\ell+k-1+i) &= N_{\mathscr{A},\mathscr{S}}(m,\ell+k+i)\\
a_0 N_{\mathscr{A},\mathscr{S}}(m,\ell+1+i) + \ldots + a_{k-1} N_{\mathscr{A},\mathscr{S}}(m,\ell+k+i) &= N_{\mathscr{A},\mathscr{S}}(m,\ell+k+1+i)\\
&\ldots\\
a_0 N_{\mathscr{A},\mathscr{S}}(m,\ell+k-1+i) + \ldots + a_{k-1} N_{\mathscr{A},\mathscr{S}}(m,\ell+2k-2+i) &= N_{\mathscr{A},\mathscr{S}}(m,\ell+2k-1+i).
\end{aligned}$$

This can be experimentally tested starting from $k=2$, and looking for the first k that satisfies these requirements (which will necessary be the smallest degree of a linear recurrence relation for the considered sequence).

Relying on [6] again reveals the following. The minimal degree of a linear recurrence relation for $N_{\mathscr{A},\mathscr{S}}(3,n)$ is 2, the minimal degree of a linear recurrence relation for $N_{\mathscr{A},\mathscr{S}}(4,n)$ is 4, the minimal degree of a linear recurrence relation for

$N_{\mathscr{A},\mathscr{S}}(5,n)$ is 8, and the minimal degree of a linear recurrence relation for $N_{\mathscr{A},\mathscr{S}}(6,n)$ is 20.

In particular,

$$N_{\mathscr{A},\mathscr{S}}(3,n+2) = 4N_{\mathscr{A},\mathscr{S}}(3,n)+7N_{\mathscr{A},\mathscr{S}}(3,n+1),$$

for all $n \geq 3$.

4 Two-dimensional linear recurrence relations

The results obtained for the noise matrices mentioned in the previous section suggest that the degree of the minimal linear recurrence relation increases with increasing number of rows. This is, however, not a universal fact concerning all matrices with prohibited bounded patterns. For example, all the numbers $N_{\mathscr{A},\mathscr{S}_2}(m,n)$ for the matrices from Example 2 are equal to 1, and hence satisfy the recurrence relation $N_{\mathscr{A},\mathscr{S}_2}(m,n+1) = N_{\mathscr{A},\mathscr{S}_2}(m,n)$. Nevertheless, we feel that the following conjecture might turn out to be true.

Conjecture 2. *Let $\mathscr{A}$ be a finite alphabet and $\mathscr{S}$ be a set of $k \times \ell$ prohibited matrices over $\mathscr{A}$ with $k,\ell \geq 2$. Then the minimal degree of a linear recurrence relation for the sequence*

$$N_{\mathscr{A},\mathscr{S}}(m,n), N_{\mathscr{A},\mathscr{S}}(m,n+1), N_{\mathscr{A},\mathscr{S}}(m,n+2), \ldots$$

$m \geq k$, increases with increasing m.

In view of Conjecture 2, instead of looking for one-dimensional linear recurrence relations, we propose to search for two-dimensional recurrence relations.

Specifically, let $r_{m,n}$ be a two dimensional sequence of reals (i.e., a function from $\mathbb{N}\times\mathbb{N}$ to $\mathbb{R}$). We say that a two-dimensional sequence $r_{m,n}$ *satisfies a two-dimensional linear recurrence relation* provided there exist coefficients $a_{i,j}$, $0 \leq i \leq t$, $0 \leq j \leq s$, with $a_{t,s} = 0$, such that

$$\begin{aligned} r_{m+t,n+s} = \\ a_{0,0}r_{m,n} + a_{0,1}r_{m,n+1} + \ldots + a_{0,s}r_{m,n+s} + \\ a_{1,0}r_{m+1,n} + a_{1,1}r_{m+1,n+1} + \ldots + a_{1,s}r_{m+1,n+s} + \\ \ldots + \\ a_{t,0}r_{m+t,n} + a_{t,1}r_{m+t,n+1} + \ldots + a_{t,s}r_{m+t,n+s}, \end{aligned}$$

for all $m,n \in \mathbb{N}$.

Our preliminary results suggest the following two conjectures.

Conjecture 3. *Let $\mathscr{A}$ be a finite alphabet and $\mathscr{S}$ be a set of $k \times \ell$ prohibited matrices over $\mathscr{A}$ with at least one of the numbers k,ℓ equal to 1. Then the two-dimensional sequence $N_{\mathscr{A},\mathscr{S}}(m,n)$, $m,n \geq 1$, satisfies a two-dimensional recurrence relation.*

Conjecture 4. *Let $\mathscr{A}$ be a finite alphabet and $\mathscr{S}$ be a set of $k \times \ell$ prohibited matrices over $\mathscr{A}$ with $k,\ell \geq 2$. Then the two-dimensional sequence $N_{\mathscr{A},\mathscr{S}}(m,n)$, $m,n \geq 1$, does not satisfy a two-dimensional recurrence relation.*

ACKNOWLEDGMENT

The authors wish to thank the referees for several useful insights that helped to improve our paper.

The first author acknowledges the support by the projects VEGA 1/0596/17, VEGA 1/0719/18, APVV-15-0220, by the Slovenian Research Agency (research projects N1-0038, N1-0062, J1-9108), and by NSFC 11371307.

The second author acknowledges the support by VEGA 1/0039/17 and VEGA 1/0719/18.

References

[1] Jajcay, R., Odhady počtu hraničných matíc. Matematické obzory, zv. **31** (1989) 41–49

[2] Goulden, I.P., Jackson, D.M., *Combinatorial Enumeration*. Wiley-Interscience Series in Discrete Mathematics, Wiley (1983)

[3] Guibas, L. J., Odlyzko, A. M., String overlaps, pattern matching, and nontransitive games. J. Comb. Theory, Ser. A **30** (1981)183–208

[4] Heubach, S., Kitaev, S., Avoiding substrings in compositions. Congr. Numerantium **202** (2010) 87–95

[5] Odlyzko, A.M., Enumeration of strings. Combinatorial algorithms on words, Proc. NATO Adv. Res. Workshop, Maratea/Italy 1984, NATO ASI Ser., Ser. F **12** (1985) 205–228

[6] Opial, M., Bounded locally testable matrices. Bachelor's thesis, Comenius University, Bratislava, 2015.

S. Krajči (ed.): ITAT 2018 Proceedings, pp. 181–187
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Tatiana Jajcayová and Zuzana Majeríková

A note on some interesting test results for the Rabin-Karp string matching algorithm

Tatiana Jajcayová, Zuzana Majeríková

Department of Applied Informatics, Comenius University, Bratislava, Slovakia
jajcayova@fmph.uniba.sk
z.majerikova19@gmail.com

Abstract: While analyzing and testing some of the well known string matching algorithms, we came across rather interesting phenomenon in tests results for the Rabin-Karp algorithm. Testing which moduli work best in the algorithm, we realized that on some texts some moduli perform much worse then others. This caught our attention, and we continued testing. In this paper, we summarize tests using different moduli on the same text, tests run on different English texts (fiction, scientific, law, newspaper), and tests run on different languages and alphabets (English, Slovak, French, German and Russian). We include analysis of the obtained results and state some hypotheses.

1 The Rabin-Karp algorithm

We start with a brief review of the Rabin-Karp algorithm and its computational complexity. This string matching algorithm is based on the use of elementary number-theoretic notions such as the equivalence of two numbers modulo a third number. Rabin and Karp [5] proposed the algorithm to improve performance of string matching algorithms in practice. This algorithm generalizes to other algorithms for related problems, such as two dimensional pattern matching.

To simplify the explanations, it will be convenient to assume for a moment that characters are decimal digits, so we interpret the characters of strings as both digits and also graphical symbols. Thus, a string of length k can be represented as a decimal number. For instatnce, a string $s =$ "12345" of length 5 corresponds to a decimal number 12 345. In general case, we assume to have an alphabet Σ which consists of characters that are digits from the number system over the base $b = |\Sigma|$.

Let us have a pattern $P[1...m]$ and text $T[1...n]$. The decimal value of pattern P will be denoted as p. Similarly, the decimal value of every substring of a text T of length m starting at a position s will be denoted as t_s. We say that pattern P occurs in text T if and only if one can find at least one t_s that has the same decimal value as p, i.e. $t_s = p$ and $T[s+1...s+m] = P[1...m]$, $s = 0, 1, \ldots, n-m$. In that case we call s a valid shift.

We can compute decimal value of p in time $\Theta(m)$ by using the Horner's Rule:

$$p = P[m] + 10(P[m-1] + 10(P[m-2] + 10(P[m-3] + \ldots$$
$$+10(P[2] + 10P[1])...)$$

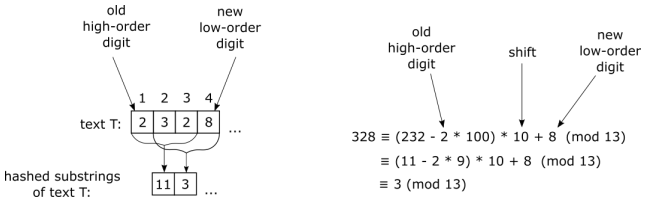


Figure 1: Computation of the value for a window based on a previous window in a constant time. The first window has a value "232". We drop the high-order digit that is "2" and multiply the rest of the window by 10. Then we add the low-order digit that is "8" and we get new value that is "328". The value of the first window is 11 and the value of the new window is 3, because all computations are performed with out hash function that is modulo 13.

Similarly, again by the Horner's Rule, the value t_0 can be computed in $\Theta(m)$.

For every s, the value t_{s+1} can be then computed from the previous value t_s:

$$t_{s+1} = 10(t_s - 10^{m-1}T[s+1]) + T[s+m+1].$$

For example, to compute the value t_1 from t_0 for our text T = "12345" from the previous paragraph and pattern P of length $m = 3$, we use the value t_0 which is 123. In this computation $s = 0$. A character at the position T[s+1] is "1", and its numerical value is 1, a character at position T[s+m+1] is "4" and its value is 4. The computation is as follows:

$$t_1 = 10(t_0 - 10^{3-1}T[0+1]) + T[0+3+1]$$
$$t_1 = 10(123 - 10^2 T[1]) + T[4]$$
$$t_1 = 10(123 - 100) + 4$$
$$t_1 = 234$$

We see that by computing $t_s - 10^{m-1}T[s+1]$, we remove the highest order digit from value t_s. By subsequent multiplication of the remainder of the value by 10, we shift the number to the left. 10 is used because we work with the decimal system. In the general case, where base b is $|\Sigma|$, we would have to shift by b. By adding $T[s+m+1]$ we add the low-order digit and we get the final value of t_{s+1}. (Figure 1)

If the constant 10^{m-1} is precomputed, then for each $s = 1, 2, \ldots, n-m$, the computation of the value t_s is done in constant time. Thus the computation of all the remaining values t_1, t_2, t_3, …, t_{n-m} is done in $\Theta(n-m)$. That

means, that by comparing value of p with every value of t_s, we can get all valid shifts in time $\Theta(m)+\Theta(n-m+1)$.

A problem may occur, when value p and values t_s are too large. Some mathematical operations with large numbers may take a long time, and the assumption that they take only constant time may be unreasonable. One possible remedy for this is, what Rabin-Karp algorithm does: instead of comparing the original values, it computes values p and t_s modulo some suitable modulus q, and then compare these new modulated numbers. The modulated numbers are not bigger than $q-1$. Computation of $p \pmod q$ takes again time $\Theta(m)$ and computation of all $t_s \pmod q$ values together takes $\Theta(n-m+1)$.

The modulus q is usually chosen to be a prime such that the value $b \cdot q$ fits within a computer word. (b is the base, the number of characters in the alphabet Σ). According to this, the previous equation for computing t_s+1 changes to:

$$t_{s+1} \equiv (b(t_s - T[s+1]h) + T[S+m+1]) \pmod q,$$

where

$$h \equiv d^{m-1} \pmod q$$

It is clear that if $p \not\equiv t_s \pmod q$ then P does not start at position s in T. So if this happens, we know that s is not a valid shift. On the other hand, it is possible that $p \equiv t_s \pmod q$, and still the pattern P does not start at s. That means that to complete this algorithm, we must not only compare the modulated values and find a match, but also test further to see if the substring $T[s+1...s+m]$ is the same as the pattern P. This additional but indispensable comparison takes another $\Theta(m)$ time. The simple graphical example of the Rabin-Karp algorithm and modulation of the values of t_s is shown in Figure 2.

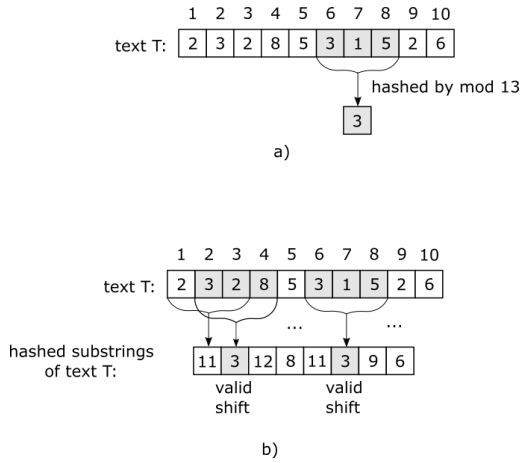


Figure 2: Example of the Rabin-Karp string matching with alphabet Σ in which every character is a decimal digit. a) an example of a hashed pattern $P = 315$ that is located in text T. Part b) demonstrates hashing of every possible substring of T with the length of P. Two substrings that have the same hash value as the pattern are found and they are denoted as valid shifts. Only the substrings with the valid shift are compared further with the pattern by characters. For this case we have found one occurrence of pattern P in text T at position 6.

In our version of the Rabin-Karp algorithm the preprocessing time takes $\Theta(n)$. The processing time takes $\Theta((n-m+1)m)$ in the worst case, and only $\Theta(n+1)$ in the best case of running. The worst case occurs when all characters of pattern P and all characters of text T are the same unique symbol. For example, if we have a pattern P="aaa" and a text T="aaaaaaaaa", then algorithm considers every possible shift as valid and behaves as the naive string matching algorithm.

2 Implementation

In this section we sketch how we implemented the Rabin-Karp algorithm. The particulars of the implementation can become interesting when discussing the test results in the last section of the paper.

We have chosen Java as the programming language. Java is an object-oriented language, and its biggest advantage is its platform independence, which means that the Java code can be run without modifying at any platform that supports Java. As a development tool we chose Eclipse Java Oxygen with a WindowBuilder component. WindowBuilder is designed to create Java GUI applications in an easy way.

2.1 Implementation of the Main Application

We study and test Rabin-Karp algorithm in a context of other string matching algorithms. To organize the testing of the implemented algorithms, we needed to create the main application with Graphical User Interface. This application is build to work with all the studied algorithms and to manage the input texts, patterns and all of the output results. The main components that we need in our application:

- **Input pattern** - text input field, where the user can write or paste the pattern that he wants to find in the text **Input text**
- **Input text** - text input field, where the user can write or paste the text where he wants to find all the occurrences of the pattern from the **Input pattern**
- **Load text from file** - button, that allows the user to load text from a file to the **Input text**
- **Algorithm** - selector, where the user can choose which algorithm he wants to test
- **Modulo** - number input field, where the user can choose the modulo that will be used in the Rabin-Karp algorithm
- **Output text** - output text field, where the text is rewritten from the **Input text**, but with highlighted occurrences of the pattern from **Input pattern**

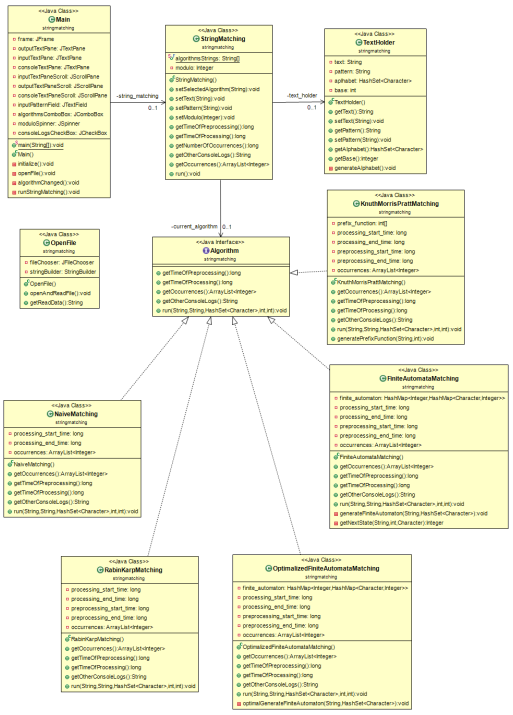


Figure 3: Class diagram of the main application generated by The ObjectAid UML Explorer for Eclipse

- **Additional console logs** - check field, if it is checked it means that in the **Console** there will be written some additional logs for the given algorithm alongside with the proccessing and preprocessing time
- **Console** - after running an algorithm, in the console there is shown it's processing and preprocessing time and when the **Additional console logs** is checked there are some additional logs for every algorithm
- **Run** - button that runs the program, where with a selected **Algorithm**, search in **Input text** for an **Input pattern**

A class diagram is shown in Figure 3.

The application consists of the following classes that work as follows:

Main is the main class that runs the application. Its purpose is to initialize the contents of the frame of GUI window. When users run the application, the Main class gets the value of the input text, the input pattern, the input modulo and the selected algorithm. The class passes all this values to class **StringMatching** and calls the function *run* of the class **StringMatching**. After the *run* is completed, the class fills the output text and console with the results of running.

TextHolder is the class that stores the text and the pattern that the application got on input. It also generates the alphabet and the base based on the given text and pattern.

OpenFile is the class that allows the user to load a text from a file with a *JFileChooser*.

Algorithm is an interface for implemented algorithms. Every algorithm needs to have implemented functions *getTimeOfPreprocessing*, *getTimeOfProcessing*, *getOccurrences*, *getOtherConsoleLogs* and *run(text, pattern, alphabet, base, modulo)*.

NaiveMatching, RabinKarpMatching, FiniteAutomataMatching, OptimalizedFiniteAutomataMatching, KnuthMorrisPrattMatching are the classes where the implemented algorithms that we used are.

StringMatching is the class that gets the selected algorithm, input text, input pattern and modulo. First, it processes the text and the pattern with **TextHolder** and gets the alphabet and the base generated from the text. Then, it creates a new instance of the selected algorithm and calls it's function *run*.

The review of the implemented application and it's GUI is shown in Figure 4, where a simple running of the program with the naive string matching algorithm with pattern "the" and text of the book "The Project Gutenberg EBook of The Adventures of Sherlock Holmes by Sir Arthur Conan Doyle" is demonstrated.

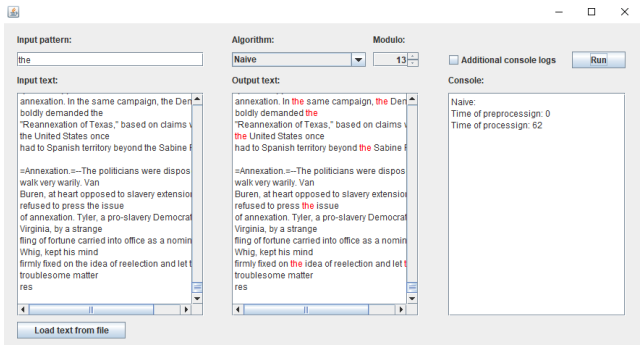


Figure 4: The main application, where the user can test the implemented algorithms.

2.2 The Rabin-Karp algorithm implementation

For the Rabin-Karp string matching procedure, after it receives the input text, the pattern, the value of d that represents the $radix - d$ notation that is uses in its matching and modulo, we need to initialize the value of the high-order digit position of an m-digit window. This is shown in Listing 1 on line 1. After that the preprocessing of this algorithm follows. We need to compute the value of p and the value of the first substring t_0 of the text with length of the pattern. Implementation of this is shown in Listing 1 on lines $3-6$, where the *pattern_hash* is our value of p and *sub_text_hash* represents the value of t_0. As a default, these variables were preset to value 0.

Listing 1: Calculation of Hashed Values of Value h, p and t_0

```
1  int h = (int) Math.pow(base, pattern_length-1) % modulo;
2
3  for (int i=0; i<pattern_length; i++) {
4      pattern_hash = (base*pattern_hash + pattern.charAt(i)) % modulo;
5      sub_text_hash = (base*sub_text_hash + text.charAt(i)) % modulo;
6  }
```

How the processing is implemented can seen in Listing 2. It starts by iterating through all possible shifts s in the text. On line 3 there is a check whether value p matches value of t_s at shift s where the loop is executed. If so, then the character by character comparison between pattern P and substring of text $T[s+1...s+m]$ follows. If these two strings are equal, we add shift to the array of all of the occurrences, where we store all shifts in which we have found a positive match. The check on line 9 will be true on every t_s, except the last one t_{n-m}. That is because the main loop will execute one more time and in that case we need to compute the value of upcoming t_{s+1}. This computation is implemented on line 10. In the case that the value of t_{s+1} is smaller that 0, we add q to this value on lines $13-15$.

Listing 2: Processing of the Rabin-Karp String Matching Algorithm

```
1  for (int s=0; s<(text_length - pattern_length); s++) {
2
3      if (pattern_hash == sub_text_hash) {
4          String text_substring = text.substring(s, (s + pattern_length));
5          if (Objects.equals(pattern, text_substring))
6              occurrences.add(s);
7      }
8
9      if (s < (text_length - pattern_length)) {
10         sub_text_hash = (base*(sub_text_hash - text.charAt(s)*h) +
11                          text.charAt(s+pattern_length)) % modulo;
12
13         if (sub_text_hash < 0)
14             sub_text_hash = (sub_text_hash + modulo);
15     }
16 }
```

3 Testing moduli in the Rabin-Karp algorithm

While testing a performance of the implemented string matching algorithms, we were interested whether a choice of the modulus q in the Rabin-Karp algorithm affects the performance of the algorithm. Recall that for the given base $b = |\Sigma|$, the modulus q is usually chosen to be a prime number such that the value $b \cdot q$ fits within a computer word. In this section we summarize results of our testing of moduli in the Rabin-Karp algorithm.

We had some preconceived ideas how the test results should look like. These expectations were met while testing some of the texts, for an example see Figure 5. The sample text in Figure 5 is an English text. It is an artificial text in the sense that it was generated in an online random text generator, with even some further random changes and additions. The text is 400 000 characters long.

A suprise came when we tested the Rabin-Karp algorithm on *The adventures of Sherlock Holmes* by Sir Arthur Conan Doyle. The text was obtained from The Project Gutenberg. Release Date: March, 1999 [EBook #1661] [Most recently updated: November 29, 2002], Edition: 12, Language: English, Character set encoding: ASCII.

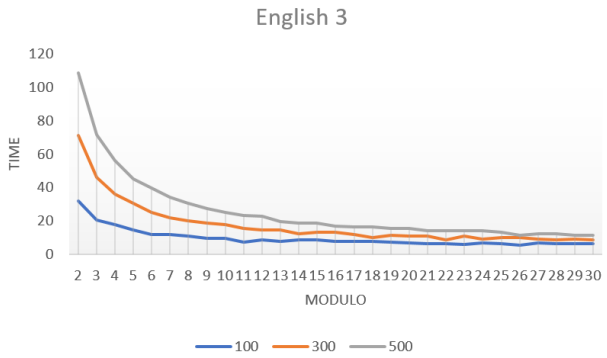


Figure 5: The Rabin-Karp algorithm on an English text generated by an online random text generator. Three different colors represent three different lengths of a searched pattern P.

The text was again about 500 000 characters long, and we tested with different patterns and different lengths of patterns. As we see in Figure 6, modulus $q = 13$ performed significantly worse than the other moduli.

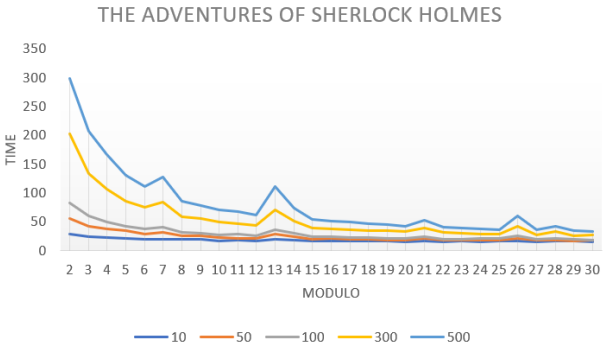


Figure 6: The Rabin-Karp algorithm on English fiction: The adventures of Sherlock Holmes by Sir Arthur Conan Doyle.

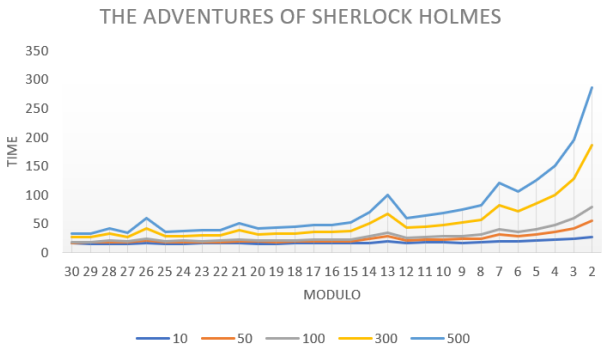


Figure 7: The Rabin-Karp algorithm on English fiction: The adventures of Sherlock Holmes by Sir Arthur Conan Doyle.

This caught our attention. We tested the same text for the second time, now with the moduli ordered in the descending order (see Figure 7) to make sure that a possible error is not caused in the program (language Java can

sometimes slow down the performance of the program because of the Garbage Collector). Since we got the two graphs that are exactly mirror images of each other, we see that the garbage collecting in Java is not the reason for the unexpected behavior.

Our first hypothesis was that the natural language has different characteristics than randomly generated texts. So we continue testing on many different types of English texts and patterns. In Figure 8, we see the results of the testing on a law text The Common Law, by Oliver Wendell Holmes, Jr., and in Figure 9 on a scientific text from the book Ancient And Modern Physics by Thomas E. Willson. (Both these text were again obtained from the Project Gutenberg.) As we see the charts for all these texts are different. To definitely show that our first hypothesis does not hold, in Figure 10 is yet another *randomly generated* English text that has $q = 13$ behaving as it is in the fiction in Figure 6.

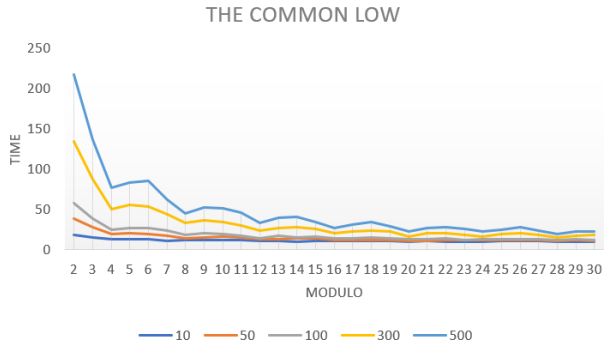


Figure 8: The Rabin-Karp algorithm on an English law text: The Common Law, by Oliver Wendell Holmes, Jr.

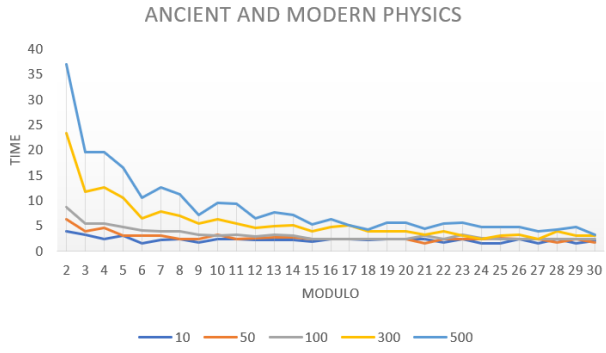


Figure 9: The Rabin-Karp algorithm on an English scientific text: Ancient And Modern Physics by Thomas E. Willson

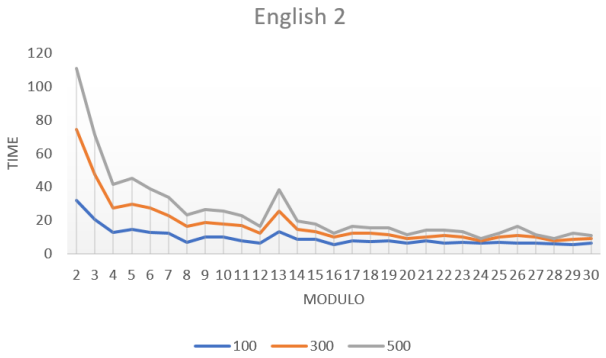


Figure 10: The Rabin-Karp algorithm on another randomly generated English text.

Our another guess was that the performance depending on moduli may somehow be related to the language or to the alphabet of the text. Therefore we tested on other languages, including German (Figure 11), and other languages using other alphabets, including Russian (Figures 12 and 13.) We see that we can get all types of behavior.

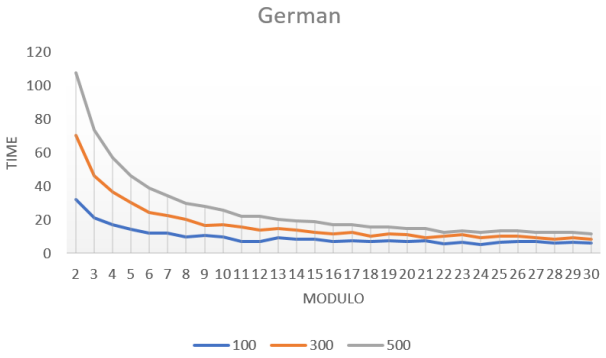


Figure 11: The Rabin-Karp algorithm on a randomly generated German text.

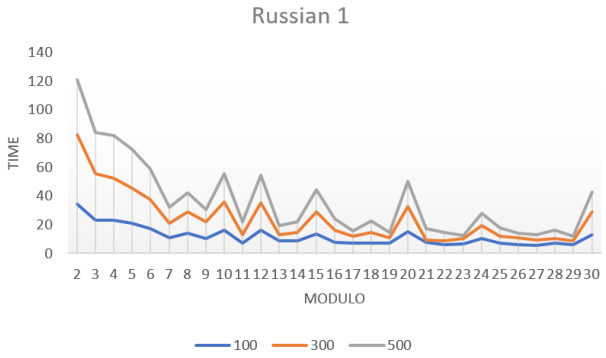


Figure 12: The Rabin-Karp algorithm on the first Russian text (randomly generated).

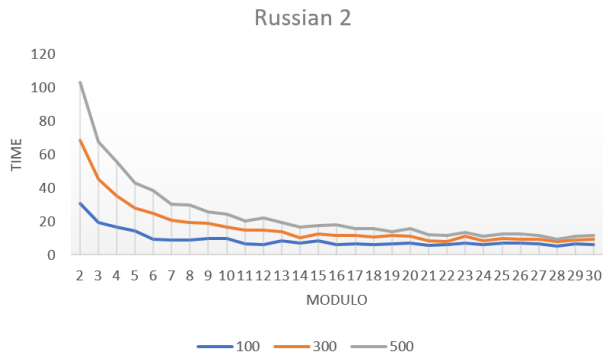


Figure 13: The Rabin-Karp algorithm on the second Russian text (randomly generated).

We were also interested what is happening in Slovak language. Below you see the results of three tests. The first one, Figure 14, is on a randomly generated Slovak text, the second one, Figure 15, is on the original Slovak fiction, Adam Šangala by Ladislav Nádaši-Jégé, and the third one, Figure 16, is on the Slovak translation of French Molier's Lakomec (L'Avare).

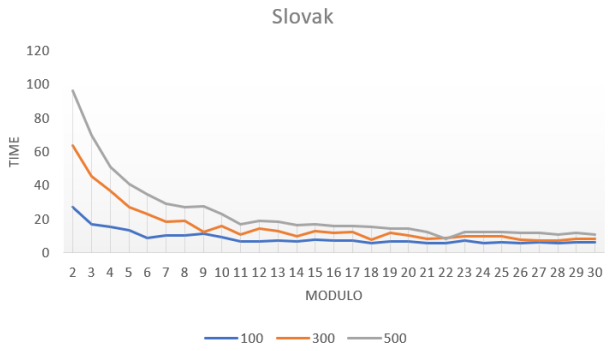


Figure 14: The Rabin-Karp algorithm on a randomly generated Slovak text.

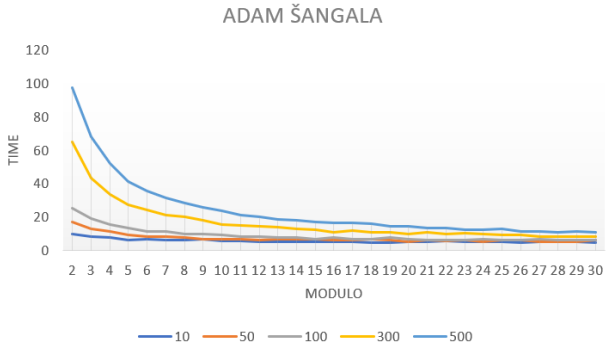


Figure 15: The Rabin-Karp algorithm on the original Slovak fiction, Adam Šangala by Ladislav Nádaši-Jégé

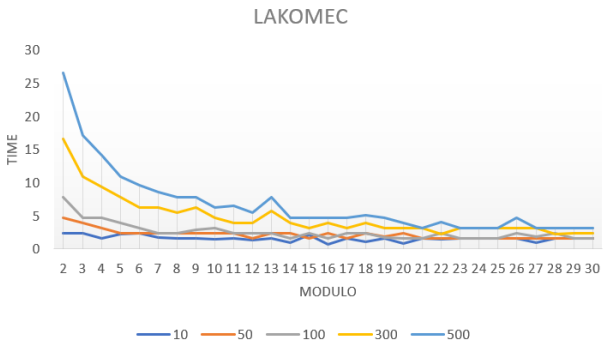


Figure 16: The Rabin-Karp algorithm on the Slovak translation of Molier's Lakomec (L'Avare).

Here it seems that $q = 13$ is again behaving worse that other moduli. (Why?) Just to make a comparison, we also include the results of the testing on the original French text. As we see below, Figure 17, modulus $q = 13$ is not different from others, while $q = 7$ and in some extend its multiples are now the moduli with the worst performance.

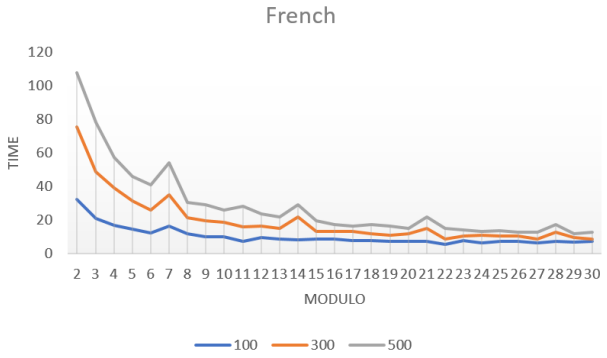


Figure 17: The Rabin-Karp algorithm on the French text.

We were hoping that by systematically running tests on different types of texts, on different languages and alphabets, with different patterns to search for, we would be able to understand the relationship between the computational time and the modulus for Rabin-Karp algorithm. At this moment we see that this tests are not sufficient to explain this relationship, and that other, more involved (probably statistical) methods would be needed to completely understand the phenomenon.

Acknowlegment

The first author acknowledges the support by VEGA 1/0039/17 and VEGA 1/0719/18

References

[1] Marc GOU, Algorithms for String matching, July 30, 2014

[2] Nimisha Singla, Deepak Garg, January 2012, String Matching Algorithms and their Applicability in various Applicationsl, International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-I, Issue-6, January 2012

[3] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, 2009, Introduction to Algorithms: Third Edition, Massachusetts Institute of Technology, Cambridge, Massachusetts 02142 <http://mitpress.mit.edu>

[4] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, 1974

[5] Richard M. Karp and Michael O. Rabin, Efficient randomized pattern-matching algorithms, IBM Journal of Research and Development, 31(2):249-260, 1987

[6] Donald E. Knuth, James H. Morris, Jr., and Vaughan R. Pratt, Fast pattern matching in strings, SIAM Journal on Computing, 6(2):323–350, 1977

[7] Edward M. Reingold, Kenneth J. Urban, and David Gries, K-M-P string matching revisited, Information Processing Letters, 64(5):217–223, 1997

[8] Zvi Galil and Joel Seiferas, Time-space-optimal string matching, Journal of Computer and System Sciences, 26(3):280–294, 1983

[9] Vidya SaiKrishna, Prof. Akhtar Rasool and Dr. Nilay Khare, String Matching and its Applications in Diversified Fields, IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 1, January 2012

Index