Czech Technical University in Prague, FIT, Dept. of Software Engineering
Charles University in Prague, MFF, Department of Software Engineering
VŠB–TU Ostrava, FEECS, Department of Computer Science
Czech Society for Cybernetics and Informatics

Proceedings of the Dateso 2014 Workshop

DATESO
Databases, Texts
Specifications, and Objects

2014

http://www.cs.vsb.cz/dateso/2014/
http://www.ceur-ws.org/Vol-1139/

April 16 – 18, 2014
Roudnice nad Labem

Technical editor:
Adam Šenk, `senkadam@fit.cvut.cz` and Pavel Moravec, `pavel.moravec@vsb.cz`

Page count:	108
Impression:	50
Edition:	1st
First published:	2014

This proceedings was typeset by PDFLATEX.
Cover design by Pavel Moravec (`pavel.moravec@vsb.cz`) and Tomáš Skopal.
Printed and bound in Prague, Czech Republic by Česká technika – nakladatelství ČVUT.

Published by Czech Technical University in Prague
FIT, Department of Software Engineering
Thákurova 9, 160 00 Prague 6, Czech Republic

Preface

DATESO 2014, the international workshop on current trends on Databases, Information Retrieval, Algebraic Specification and Object Oriented Programming, was held on April 16 – 18, 2014 in Roudnice nad Labem.

The 14th year was organized by Department of Software Engineering, FIT ČVUT Praha, Department of Computer Science VŠB-Technical University Ostrava, Department of Software Engineering MFF UK Praha, and Working group on Computer Science and Society of Czech Society for Cybernetics and Informatics. The DATESO workshops aim for strengthening connections between these various areas of informatics.

The proceedings of DATESO 2014 are also available at DATESO Web site: `http://www.cs.vsb.cz/dateso/2014/` and CEUR Workshop Proceeding site: `http://www.ceur-ws.org/Vol-1139/` (ISSN 1613-0073). The Program Committee selected 9 papers (6 full and 3 short papers) from 11 submissions, based on two independent reviews.

We wish to express our sincere thanks to all the authors who submitted papers, the members of the Program Committee, who reviewed them on the basis of originality, technical quality, and presentation. We are also thankful to the Organizing Committee Special thanks belong to Czech Society for Cybernetics and Informatics.

Our thanks go also to Pavel Moravec who, as copy editor of DATESO Proceedings, helped to prepare this volume and provided technical support for the conference preparation portal.

April, 2014 J. Pokorný, K. Richta, V. Snášel (Eds.)

Table of Contents

Towards a Harmonic Complexity of Musical Pieces

Ladislav Maršík[1], Jaroslav Pokorný[1], and Martin Ilčík[2]

[1] Dept. of Software Engineering, Faculty of Mathematics and Physics
Charles University, Malostranské nám. 25, Prague, Czech Republic
{marsik, pokorny}@ksi.mff.cuni.cz
[2] The Institute of Computer Graphics and Algorithms,
Vienna University of Technology, Favoritenstraße 9-11, Vienna, Austria
ilcik@cg.tuwien.ac.at

Abstract. Music information retrieval (MIR) is small, but fast growing discipline. It aims at extraction of relevant information from music in order to improve the work with multimedia information systems. A popular approach in MIR is to apply signal processing techniques to extract low-level features from a musical piece. However, in order to create helpful applications, high-level concepts, such as music harmony, need to be examined as well. In this paper, we present a new model for understanding music harmony in computer systems. The proposed model takes the challenge of filling the gap between music theory and mathematical structures. Inspired by the computational complexity, we then derive a new term, harmonic complexity, that can be evaluated for a musical piece, and we test it on a genre classification problem.

Keywords: music information retrieval, harmonic analysis, harmonic complexity, chord transcription, chord progression

1 Introduction

A massive distribution of digital media has made music information retrieval (MIR) a wanted field of study. Musicians and non-musicians both benefit from applications that ease their work with music, such as notation software or internet radios. One of the recent challenges has been to develop an intelligent retrieval of musical piece based on user preferences (Schönfuss [13]). However, information systems are still not able to understand the music in its full depth and more research is needed in this field. In this work, we narrow our focus on content-based recommendation and classification tasks. Rather than giving a complete solution, we define a new term, harmonic complexity, and a model of music harmony, to help future research in both tasks, and provide tools for other harmony-related problems.

Content-based recommendation or classification can be addressed using rules of music acoustics, by computing the Discrete Fourier Transform (DFT), and dealing with the frequency-domain features (Li [8]). To achieve even more accuracy, some knowledge of higher-level concepts can be used, such as chord progression or key analysis (Absolu et al. [1], Sapp [11]). That leads us to the use of music theory and tonal harmony.

Tonal harmony. Music theory is a highly developed field that provides us with a framework for working with musical structures. In our work, we choose music theory, and in particular, tonal harmony (TH), to help us understand one of the important aspects in music. Our focus on tonal harmony comes from understanding, that acoustic sounds (tones) sounding simultaneously form structures, which even listeners without explicit musical training implicitly recognize (Krumhansl [4]). Brief definitions of TH can be found in Section 2.

Harmonic complexity. With music classification and recommendation tasks, naturally a question arises: What features can we use to determine the music genre or to match the music with user's preference? There have been several attempts to select the relevant musical features resulting in useful applications (Schönfuss [13] or Pandora music radio[3]). However, in the most successful systems like Pandora, the extraction of relevant features is still a domain of human analysis. We propose a new feature, harmonic complexity, that simulates the process a trained musician would use to analyze the musical piece. Whenever the music obeys simple TH rules, we assign it a lower value of complexity, whereas if the rules are complex, or the harmony does not obey any known rules, we assign it higher values. For this purpose, we have created a model of harmonic complexity based on formal grammars.

We are motivated by the fact, that although TH is a stable theory based on music acoustics (e.g. Schönberg [12]), there is still a gap between the formalizations of music theory and mathematics. We believe that new models can clarify the connection between the fields. Moreover, TH provides us an universal approach to analyze music, which is independent from any genre or any given dataset, as opposed to machine learning.

To summarize, the main contributions of this paper are:

- Defining a new term, harmonic complexity, that can be used as a new musical feature to aid the content-based music recommendation and music classification tasks
- Proposing a mathematical model based on tonal harmony, reusable for future harmony-related tasks
- Testing the proposed model and harmony complexity evaluations on a music classification problem

After providing definitions of TH, we summarize the results of the related work in Section 3. We then describe our model of harmonic complexity in Section

[3] http://www.pandora.com

4. Afterwards, we propose its usefulness for music classification challenge in Section 5. In the end we provide conclusion and discuss the future work, as well as the connection of our research to music recommendation task.

2 Tonal harmony

In this section we provide brief definitions of TH, based on Schönberg [12] and Riemann [10]. We consider the following terms to be known to the reader and reference Schönberg [12] or Krumhansl [5] for clarification: *tone*, *octave*, *accidentals* (*sharp* ♯ or *flat* ♭), *semitone* and *whole tone* intervals, *chord*, *chord root*, *scale*, *key* and *degrees* of the scale or key.

We further make use of the concept of *basic harmonic functions* characteristic for each key. The harmony in music usually starts in the *tonic*, a function of harmonic steadiness and release. Optionally, it deviates to the *subdominant*. Finally, the harmonic movement culminates in the *dominant*, a function representing the maximal tension, requiring a transition back to the tonic. According to Zika [16] it is the skeleton of every music motion in musical pieces in the TH system.

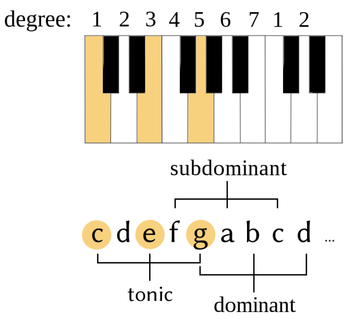

Fig. 1. Basic harmonic functions in the *C major* key

In addition to the three basic harmonic functions, there are also variants, or parallels of the basic harmonic functions. We provide a simplified definition of a function parallel based on the original definition by Riemann [10].

Definition 1. Function parallel *is the chord created from the basic harmonic function either by extending the highest tone (i.e. fifth degree in the tonic, first degree in the subdominant and second degree in the dominant) by a whole tone, or by diminishing the root tone by a semitone. We denote a parallel by adding a „P“ subscript to the function* ($T \rightarrow T_P$)

An example parallels for the *C major* chord *c*, *e*, *g* are *c*, *e*, *a* or *b*, *e*, *g*.

We also add a definition from one of the modern interpretations of TH, described by Volek [14] and based on Leoš Janáček's conception of added dissonances to the chord. We will use it in accordance with the original concept.

Definition 2. Chord with an added dissonance *is a chord enriched with tones that did not originally belong to the chord (non-chord tones), thus creating a whole tone or a semitone dissonance.*

An example of a chord with an added dissonance is c, e, f, g, with the tone f added to the original *C major* chord c, e, g.

Some well-known tunes have a simple harmonic structure, for example *Happy Birthday*: T – D – T – S – T – D – T. However, more complex compositions, such as Smetana's *Vltava* from *Má vlast* can be interpreted as: T – S_P (D) – T_P (T) – S – T – D – T. We can notice the use of the function parallels. Moreover, some functions can have multiple meanings, if we consider switching to a different key (functions in the parentheses). We capture this principle in the next section.

3 Related work

In this section we provide the summary of the work most related to ours.

De Haas, Magalhães, and Wiering [2] have described, how music harmony analysis can improve chord transcription algorithms (detecting the correct chord progression for a musical piece). The authors have found statistically significant improvement, when the tonal harmony analysis was used. The presented Haskell-based system HarmTrace[4] is capable of deriving a tree structure explaining the tonal function of the chords in the piece. The number of errors in creating the tree can be considered as a possible way to calculate the harmonic complexity using tonal harmony, even though it was not the aim of the work.

Lots of works have been done on *tonal tension* (see Lerdahl and Krumhansl [7]). Tonal tension focuses on the distance from the tonic in every moment and so describes the harmonic movements of a musical piece. However, speaking about complexity, we may want to consider tonic, subdominant, and dominant, all three as the fundamental parts of a musical piece with a simple harmony structure.

Finally, the works on *chord distance* are closely related to our research. In his inspiring work, Lerdahl [6] introduces the term *tonal pitch space*, a model describing distances between pitches, chords, and keys. The model of a space starts with the layer of semitones, upon which other four layers are built. The number of transformations of the basic space measure the distance between the chords.

If new concepts are to be designed, they need to be in accordance with previous research.

[4] http://hackage.haskell.org/package/HarmTrace-2.0

4 Model of harmonic complexity

The basic idea of our model is simple: The use of basic harmonic functions does not increase harmonic complexity. However, every single use of a parallel or added dissonances increase the complexity of a musical piece. We can recognize the use of complex harmonies by evaluating each pair of succeeding chords – evaluating their harmonic *transition*. If we shape the three basic transitions (between T, S, D) in a triangle, the idea of increasing complexity can be illustrated as in the Figure 2.

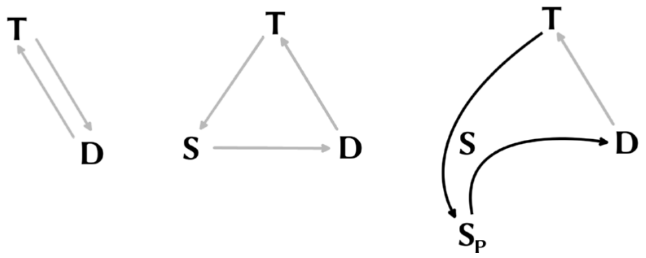

Fig. 2. Increasing complexity of a harmonic movement: The transitions T – D – T or T – S – D – T are assigned zero complexity, but the transitions T – S_P and S_P – D are considered more complex.

We base our model on the overtone series and consequent harmony rules by Schönberg [12] and Riemann [10]. We further chose not to differentiate between the transitions $S - D$, or $D - S$, and the rest of the transitions between the basic harmonic functions. In particular, the transition $D - S$ violates the rules of the original TH. However, there are some exceptions described by Zika [16] and the mentioned transition is appearing more often in today's music. Moreover, we chose the modifying of the basic harmonic functions to be our primary target in evaluating the harmonic complexity. The transitions between the basic harmonic functions itself are principally different from creating parallels or adding dissonances and therefore should not be considered complex in this concept.

We also merge the concepts of a *function parallel* and a *chord with an added dissonance* defined in Section 2, since both are a way to modify the basic harmonic function. The aim here is to generalize the process of creating complex harmonies, while focus on specific aspects will be a subject of our future work.

4.1 Formalization of transitions

We now want to evaluate the transition from the basic harmonic function to its parallel (e.g. $T \rightarrow T_P$) and in between the parallels (e.g. $T_P \rightarrow S_P$). That alone can give us information how much the music deviates from the simplest progression and translates to harmonic complexity. We use a linguistic approach to evaluate these transitions.

First of all, let us consider the simplest case: $T \rightarrow T_P$. We wish to create a parallel either by adding a tone or modifying a tone. Formally, we define a *sentential form* as a form of chord notation and two different parametric rules to modify the sentential form: *add(t)* and *alter(t,alt)*.

Definition 3. *A chord is written in* sentential form $t_1t_2...t_n$ *if it consists of the tones* $t_1, t_2, \dots, t_n$*, and for* $i \neq j$*:* $t_i \neq t_j$*, moreover* $t_1 \dots t_n$ *are ordered in the order of the chromatic scale.*

The sentential form is therefore only an ordered enumeration of chord's tones, neglecting the duplicity (working in one octave). We further define the transition rules between the sentential forms in the same fashion as Hopcroft, Motwani, and Ullman [3], but with notable differences. We design our own types of rules, applicable to the whole sentential form. We also try to avoid excessive formalism. In the following we assume, that the derivation starts with the sentential form representing the (original) basic harmonic function – an analogy with the start symbol of Hopcroft et al.

Definition 4. *The* add(t) *rule between two sentential forms h and g is defined as follows:* $h \xrightarrow[add(t)]{} g \Leftrightarrow (h = t_1, t_2, \dots t_n) \wedge (g = t_1, \dots t_j, t, t_{j+1}, \dots t_n)$*, where* t *belongs to the same key as* $t_1 \dots t_n$*, and* $\forall i : t \neq t_i$*.*

We wish to simulate both adding dissonances and creating Riemann's parallels. We choose an approach which can be confusing at first – both transformations can be achieved using the add rule. If we set, that the basic harmonic function that we start with does not necessarily need to contain 3 tones, but possibly also a single tone or two tones (substituting the basic harmonic function), using add rule we can indeed get a Riemann's function parallel. For example, we can get a parallel *cea* as *ce* + *a*, where *ce* substitutes the tonic *ceg* in *C major*. The alter rule will therefore have a different role – moving the tone outside the key, and so creating sharp dissonances and alterations in the way that TH describes them [12].

Definition 5. *The* alter(t,alt) *rule between two sentential forms h and g is defined as follows:*

$$h \xrightarrow[alter(t,alt)]{} g \Leftrightarrow (h = t_1, \dots\dots t_i, t, t_{i+1}, \dots t_n) \wedge (g = t_1, \dots t_i, t_{alt}, t_{i+1}, \dots t_n),$$

where t *belongs to the same key as* $t_1 \dots t_n$*,* t_{alt} *does* not *belong to that key, and* t_{alt} *is created from* t *by augmentation (*$alt = \sharp$*) or diminution (*$alt = \flat$*) by a semitone. Moreover,* t *can not be one of the tones of the original basic harmonic function of* h *(the basic harmonic function that the whole derivation started with).*

The constrains we cast on the add and alter rules yield the rules of TH. First of all, we can not modify the original tones, because we do not want to lose the sense of the basic harmonic function. If we wish to weaken the basic harmonic functions, we always have the possibility to start with only a subset if its tones. Secondly, we can add the tones, but only from the same key. And thirdly, we can alter the tones to a different key, but only after they have been added. Such hierarchy agrees with Lerdahl's definition of tonal pitch space [6].

Note that, using add and alter rules we are able to create arbitrary harmonies. The only thing we need to keep in mind is, that given a sentential form h, we first have to find out, which key h belongs to. There are 24 possibilities, comprising 12 major and 12 minor keys. Then we can apply the add and alter rules.

Remark 1. In a simple example we derive the tonic parallel (T_P) $cef\sharp g\sharp$ from the original tonic (T) ce, in the key *C major*: $ce \xrightarrow[add(f)]{} cef \xrightarrow[alter(f,\sharp)]{} cef\sharp \xrightarrow[add(g)]{} cef\sharp g \xrightarrow[alter(g,\sharp)]{} cef\sharp g\sharp$.

We first define a chord complexity for a simple chord. Then we proceed to define a transition complexity between two chords.

Definition 6. *A* chord complexity *(in the given key)* $c(h)$ *of chord* h *is the minimal length of derivation of the chord's sentential form (in the given key).*

Remark 2. For our example tonic parallel: $cef\sharp g\sharp$, the chord complexity in the key *C major* is 4. However, overall chord complexity for this chord can be even lower. If we choose the key *G major* and treat the form *ce* as a subdominant in *G major*, we will have 3 as the resulting chord complexity (due to the fact that the tone $f\sharp$ can be added in one step – it is a part of the key *G major.*

Understanding that the chord h can be treated in multiple keys, have multiple derivations and multiple chord complexities, we always use the shortest derivation (lowest complexity) to define the chord. We label such chord complexity $c(h)$. We use similar approach in our final definition – chord transition complexity.

Definition 7. *Let us have the sentential form of chord h_1 and the sentential form of chord h_2. We define a* chord transition complexity $tc(h_1, h_2)$*. We differentiate between these possibilities:*

1. *Let us assume, that there is a common ancestor in both derivations of h_1 and h_2 – a sentential form a, that h_1 and h_2 can be both derived from, h_1 in k_1 steps, and h_2 in k_2 steps. Then if $k_1+k_2 \leq c(h_1)+c(h_2)$, the chord transition complexity $tc(h_1, h_2)$ equals to k_1+k_2. Otherwise, $tc(h_1, h_2) = c(h_1)+c(h_2)$.*
2. *Let us assume, that there is no such common ancestor. $\exists r_1, r_2, k_{h1}, k_{h2} : h_1$ can be derived from an original basic harmonic function r_1 in k_{h1} steps, h_2 can be derived from an original basic harmonic function r_2 in k_{h2} steps.*
   (a) *Let us assume that r_1, r_2 are such basic harmonic functions, that are from the same key and the sum $k_{h1} + k_{h2}$ is minimal. Then the chord transition complexity $tc(h_1, h_2)$ equals to $k_{h1} + k_{h2}$.*
   (b) *The common key for r_1 and r_2 does not exist. Then the chord transition complexity $tc(h_1, h_2)$ equals to $u_1 + u_2$, where u_1 is the number of steps required to derive h_1 from the empty sentential form, and u_2 is the number of steps required to derive h_2 from the empty sentential form.*

In other words, we define the transition complexity between two chords as the amount of steps needed to disassemble the first chord into its basic harmonic function (rolling back the derivation), then we switch the function and count the number of steps to assemble the new chord. The resulting number of steps is the transition complexity. If there is a common ancestor in derivations, closer than the basic harmonic function, we disassemble and assemble only to and from this common ancestor. Finally, if no common key can be found for the two harmonies, we choose to disassemble the first chord all the way to an empty sentential form, and construct the second chord from an empty sentential form.

In all of these cases, it is only a simple constructing and destructing the chords, using our derivation rules. This is when the analogy with computational complexity comes in – the chord transition complexity can be understood as the computational complexity of reconstructing the chord h_1 to h_2. The illustration of chord transition complexity can be found in the Figure 3.

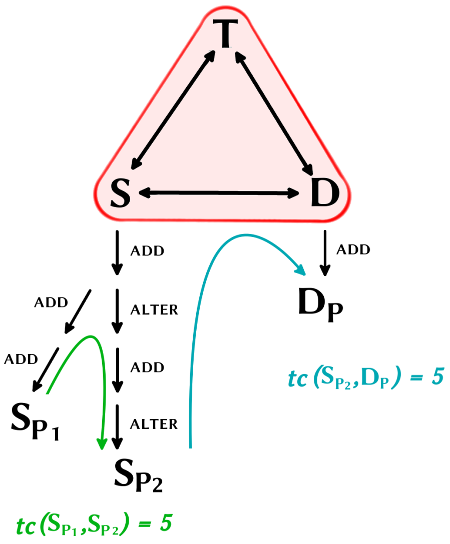

Fig. 3. Chord transition complexity: The chords S_{P1} and S_{P2} have a common ancestor in the derivation, and are assigned a chord transition complexity of 5. The chords S_{P2} and D_P do not have a common ancestor in derivation, therefore their chord transition complexity is equal to the sum of their chord complexities.

4.2 Harmonic complexity of a musical piece

Finally, we can proceed to evaluation of harmonic complexity for a whole piece. We have a musical piece M. We can use different algorithms (some of them are described in the Section 5) to extract the sequence of chords $\{C_i\}_{i \leq l}$ of the length l. Keeping in mind the analogy with the computational complexity, we are interested in obtaining an average number of steps needed to construct a chord from the basic harmonic functions. We average the values because we want to abstract from the length of the progression, in the same fashion as the computational complexity theory abstracts from the length of the input. Other interesting statistical values for harmonic complexity will be the subject of our future work.

Definition 8. *An* average transition complexity (ATC) *for a musical piece M, a sequence of chords $\{C_i\}_{i \leq l}$ of the length l and a sequence of its transition complexities $\{t_i\}_{i \leq l-1}$ is defined as follows:*

$$ATC(M) = \frac{\sum\limits_{i=0}^{l-1} t_i}{l-1}$$

4.3 Implementation methods

Understanding that, for a given chord, multiple basic harmonic functions can be found, along with multiple different keys, the model can become difficult to implement. We propose two methods of implementation, based on internal representation of musical data:

1. Query method – The fundamental rules of TH (keys, basic harmonic functions) are saved in a database and a number of queries precedes the calculation of transition complexity for chords.
2. Graph method – We can also abstract from the ambiguity of keys and functions and avoid complicated queries. Working within the bounded space of all possible sets of tones in the octave, we have designed our derivation rules in such fashion, that for a given chord it is easy to calculate all chords with transition complexity of 1 from a given chord. That approach can result in constructing a graph representation of the problem, with edges representing a transition complexity of 1. On such a graph, queries can be easily implemented using the breadth-first search algorithm.

5 Experiments

In our experiments, we focused on determination, whether the newly defined concept of harmonic complexity can be a good descriptor for music classification task. The aim is to show that ATC values (see Definition 8) can distinguish the different genres, artists in one genre, or even the songs from the same artist.

The dataset was selected from the top 5 best-selling artists in 2013, according to renowned worldwide music charts and separate for each genre: Rock and Pop[5], Jazz[6] and Classical music[7]. In each genre, 25 different pieces were analyzed.

For obtaining a chord sequence, Vamp-plugins NNLS Chroma and Chordino[8] version 0.2.1 by Mauch [9] were used. We used Chordino plugin to find where the harmony significantly changes. Then we used NNLS Chroma plugin to obtain the tones sounding in each time interval. For the purpose of these experiments, we have selected the 4 most significant tones in each interval to form a chord representation, allowing (multiple) dissonances.

We split the experiments into 3 parts.

1. First we measure the mean ATC for different genres. Since the Classical music developed historically and particular music periods are known to obey certain composition styles, we show the same type of analysis for music periods of Classical music.
2. Second we measure the mean ATC for different artists from the Rock genre.
3. Third we measure the ATC found for different songs selected from the 2 artists of Rock music: *The Beatles* and *Queen*.

The Figure 4 shows the result of the analysis. ATC values around 1 means that the transitions were not complex and aligned with the basic tonal harmony theory. On the other hand, values above 3 meant that the transitions needed on average more than 3 steps. We can see how our model gives the highest ranking to the Jazz music. Rock music averaged around 2 and the lowest rankings were assigned to Pop songs. This aligns with the expectations. There is an ambiguity between Rock and Pop genres, that is understandable, because the genres may overlap.

In Classical music periods we can observe how the music has developed through the centuries. This also corresponds with the theoretical knowledge, knowing that after Romanticism, the composers began to break the established harmony rules.

We have also discovered some interesting results about the concrete artists and songs. Queen songs were analyzed to be considerably more complex than the rest of the Rock artists, that can be attributed to their famous ensembles sounding together in an unusual way. Some particular songs have the ATC value that is out of the area where the songs of the same genre belong to with high probability (highlighted). In one particular case (*Radio Ga Ga* by *Queen*), we can conclude that, in correspondence with our results, the harmonic movements of the song are really considered to be less complex, since the song is known to have more popular genre than the rest of *Queen*'s production. The song *Let It Be* by *The Beatles* is also known for its repetitive chorus containing 4 basic chords.

[5] Source: http://www.billboard.com
[6] Source: http://www.artistsdirect.com
[7] Source: http://www.classical-music.com
[8] http://isophonics.net/nnls-chroma/

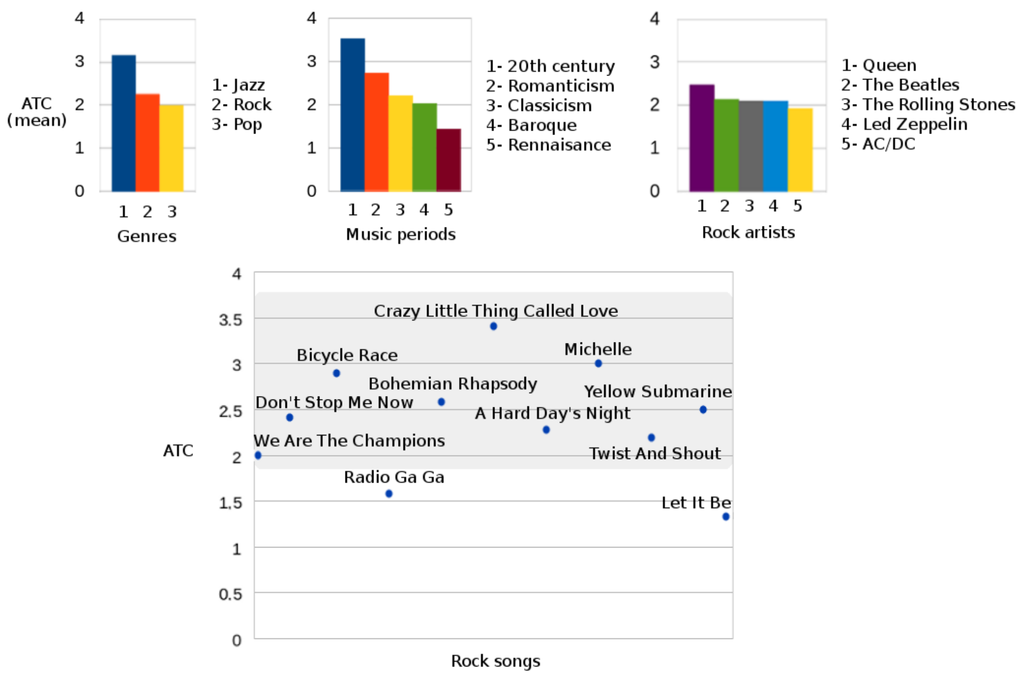

Fig. 4. Mean ATC for genres, music periods, Rock artists and ATC for individual Rock songs. The grey interval marks the typical complexity of Rock songs.

6 Conclusion and discussion

In this paper we have proposed a new term that can be evaluated for a musical piece – harmonic complexity. We have presented a mathematical model based on tonal harmony, for evaluating harmonic complexity. Lastly, we have successfully conducted a series of experiments to prove that the new feature is relevant for music classification task. The results correspond with the expectations. Knowing that our model can evaluate the harmonic movements precisely we propose using this technique to obtain music descriptors for future music classication attempts.

In the discussion we would also like to propose an idea of content-based music recommendation based on harmonic complexity. As Zanette pointed out [15], the complexity and change in music together with the repetition of pleasant stimuli are two fundamental, but contradictory, principles that the listeners are looking for in music. That translates to the initial idea that drives our research – the expectations of music listeners are different and subjective. One listener may prefer simple harmonies, while other might prefer more complex or more specific music (e.g. jazz or modern classical music). The model of harmonic complexity therefore presents a new way of understanding the need of listeners and can be used to recommend him music according to his/her needs.

For future work, we propose music classification experiments with the use of harmonic complexity as one of the music features. We also realize that harmonic complexity as described is only one of the possible implementations of the broad term of complexity in music and much can be done in specifying other similar measures (space complexity, complexity of modulations, the use of seventh chords, etc.). We hope that by showing this point of view we have induced some new ideas for the future research.

Acknowledgments. The study was supported by the Charles University in Prague, project GA UK No. 708314.

Bibliography

1. Absolu, B., Li, T., Ogihara, M.: Analysis of Chord Progression Data. In: *Advances in Music Information Retrieval*, *Studies in Computational Intelligence*, vol. 274. Springer (2010)
2. De Haas, W.B., Magalhães, J.P., Wiering, F.: Improving Audio Chord Transcription by Exploiting Harmonic and Metric Knowledge. In: *Proceedings of the 13th International Society for Music Information Retrieval Conference.* ISMIR 2012 (2012)
3. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Boston, second edn. (2001)
4. Krumhansl, C.L.: The Cognition of Tonality – As We Know It Today. Journal of New Music Research 33/3 (2004)
5. Krumhansl, C.L.: The Geometry of Musical Structure: A Brief Introduction and History. Computers in Entertainment 3/4 (2005)
6. Lerdahl, F.: Tonal Pitch Space. Oxford University Press, Oxford (2001)
7. Lerdahl, F., Krumhansl, C.L.: Modeling Tonal Tension. Music Perception: An Interdisciplinary Journal 24/4 (2007)
8. Li, T., Ogihara, M., Li, Q.: A Comparative Study on Content-based Music Genre Classification. In: *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval.* SIGIR '03, ACM (2003)
9. Mauch, M., Levy, M.: Structural Change on Multiple Time Scales as a Correlate of Musical Complexity. In: *Proceedings of the 12th International Society for Music Information Retrieval Conference.* ISMIR 2011 (2011)
10. Riemann, H.: Harmony Simplified. Augener Ltd., London (1896)
11. Sapp, C.S.: Visual Hierarchical Key Analysis. Computers in Entertainment 3/4 (2005)
12. Schönberg, A.: Theory of Harmony. University of California Press, Los Angeles (1922)
13. Schönfuss, D.: Content-Based Music Discovery. In: *Exploring Music Contents*, *Lecture Notes in Computer Science*, vol. 6684. Springer (2011)
14. Volek, J.: The Structure and Figures of Music. Panton, Prague (1988)
15. Zanette, D.H.: Music, Complexity, Information. The Computing Research Repository 0807.0565 (2008)
16. Zika, P., Kořínek, M.: Tonal Harmony for 1st-3rd Class of Music Conservatory. SPN, Bratislava (1990)

INTLIB – an INTelligent LIBrary*

Irena Holubová[1], Tomáš Knap[1], Vincent Kríž[2], Martin Nečaský[1], and Barbora Vidová-Hladká[2]

[1] Department of Software Engineering
Faculty of Mathematics and Physics, Charles University in Prague
Malostranské nám. 25, 118 00 Praha 1, Czech Republic
holubova@ksi.mff.cuni.cz
[2] Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics, Charles University in Prague
Malostranské nám. 25, 118 00 Praha 1, Czech Republic
hladka@ufal.mff.cuni.cz

Abstract. In this paper we describe the project *INTLIB – an INTelligent LIBrary* whose aim is to provide a more sophisticated and user-friendly tool for querying textual documents than full-text search. On the input we assume a collection of documents related to a particular problem domain (e.g., legislation, medicine, environment, etc.). In the first phase we extract from the documents a *knowledge base*, i.e. a set of objects and their relationships, which is based on a particular ontology (semantics). In the second phase we deal with sophisticated and user friendly visualization and browsing (querying) of the extracted knowledge. The whole system is proposed as a general framework which can be modified and extended for particular data domains. To depict its features we use the legislation domain.

1 Introduction

Nowadays, large collections of documents form one of the main sources of information and their sophisticated browsing or querying is the key aspect in many areas of human activity. Existing solutions to the problem of searching large collections of documents typically implement two approaches. The *full-text search* allows the user to find documents with the highest frequency of occurrences of a specified set of keywords. The search is automatically optimized using a pre-generated index that keeps track of the occurrences of keywords. Other approaches enable to search, e.g., the co-occurrence of words, specify their proximity, etc. By contrast, the *metadata search* allows the user to find documents with given properties (such as, e.g., author, creation date, expiration date, list of keywords, etc.). Nevertheless, the metadata are assigned to the documents manually and, thus, inefficiently and expensively.

In general, both the common approaches do not work with the *semantics* (meaning) of the documents in the collection. For example, considering the legislation, we may need to know that the term "the High Court" means a particular institution in a particular country that has certain powers and relations to the Constitutional Court. To enable the user to access the data this way means:

* This work was supported by the TAČR grant INTLIB, no. TA02010182.

1. to interpret the semantics of the documents in terms of real-world objects and the relationships between them which are described in the documents,
2. to transform the interpretation into a suitable database preferably having a standard format and standard query language, and
3. to present the interpretation to the user in a form which enables sophisticated, precise and user-friendly browsing and filtering.

In this paper we describe project *INTLIB – an INTelligent LIBrary* whose aim is to provide a more sophisticated and user-friendly tool for querying textual documents than full-text or metadata search. On the input we assume a collection of human-written documents related to a particular problem domain. INTLIB processes the data in two phases. In the *extraction phase* we extract from the documents a *knowledge base*, i.e. a set of objects and their mutual relationships, which is based on a particular ontology. The extraction phase first exploits and utilizes linguistic approaches and machine learning techniques. Then it applies algorithms for cleaning and linking of the data, and their transformation to RDF [9]. In the *presentation phase* we deal with efficient and user-friendly visualization and browsing (querying) of the extracted knowledge. The whole system is proposed as a general framework which can be modified and extended for various data domains using plug-ins. Naturally, each of the domain may require specific features; however, the general methodology we propose will remain the same. To depicts the features of the framework we use the legislation domain and we implement plug-ins that process the legislation of the Czech Republic.

The rest of the paper is structured as follows: In Section 2 we provide a larger motivating example from the area of legislation. In Section 3 we describe the current systems used for legislation processing in the Czech Republic and in Section 4 solutions used abroad. In Section 5 we propose the architecture of INTLIB and describe particular modules. In Section 6 we conclude and outline future work.

2 Motivating Example: Legislation

Sources of law are usually structured into *sections* which may contain further subsections. Moreover, a source of law may contain links to other sources which may target not only a whole source of law but also its particular section. Therefore, the structure encoded in sources of law and links between them form a complex network which the users want to browse and search for relationships between sources of law and/or their parts. Common use cases are, e.g.:

– A user is reading a particular section of an act. He would like to see what court decisions have been made in the last decade related to this particular section.
– A user is working with a particular amendment. He would like to see what sections of what acts have been corrected by this amendment or by its section.
– A user is reading a particular section of an act. He would like to find out what amendments correcting the chosen section will come to force in the next year.

A natural solution is to enable machines to search for the relationships. However, much has to be done to achieve an efficient software solution. In particular, we have to find out ways of how to:

- automatically extract the logical structure of sources of law,
- assign unique machine interpretable identifiers to the sources of law as well as to their sections and subsections so that they can be linked,
- automatically extract the links between sources of law and their parts,
- represent the extracted structure and links in a data format suitable for representing generic graph structures.

In INTLIB we concentrate on both recognizing the logical structure of source of law and recognizing references (links) between them automatically in their textual representations. We also propose a data structure which allows us to represent the recognized structure and links in a way suitable for further database processing.

Besides the logical structure and links, sources of law contain also semantic information. This is mainly the case of acts (and their amendments). Acts and other sources of law define *rights* and/or *obligations* of *natural* and *legal persons*. Different sources of law define different rights and obligations for the same kind of natural or legal person or for different persons which are, however, semantically related (e.g. one person is a special type of another person and it "inherits" the rights and obligations). Therefore, the rights and obligations of persons defined by acts and other sources of law form a complex network, similar to the described network of links among sources of law. In this case the network is defined by the semantic information encoded in the sources of law and we can therefore speak about a *semantic network* or a *knowledge graph*. Again, it would be useful for users to be able to browse and query such network. We list some sample common use cases and demonstrate them in Figure 1:

- A user wants to know what are the obligations of his employer regarding his health insurance. For example, according to the sample network depicted in Figure 1, the user can get information that his employer has an obligation to record employee's documentation, notify insurance company about changes in case of changes in employee's information, etc.
- A user wants to know what kind of information his health assurance company has to provide him. For example, according to Figure 1, the user can see that he has the right to obtain information from his insurance company about services provided and paid by the company as well as information about prices of services which are paid by him.

A software solution which enables browsing the network of the semantic concepts and relationships and automates searching and querying the network and its visualizations would be helpful for users. However, it is again necessary to solve various problems, such as to:

- automatically extract the semantic concepts and relationships between them from the textual representation of sources of law,
- assign unique machine interpretable identifiers to the concepts so that they can be linked on each other and other extending information can be linked on them,
- represent the extracted concepts and links between them in a data format which allows further database processing.

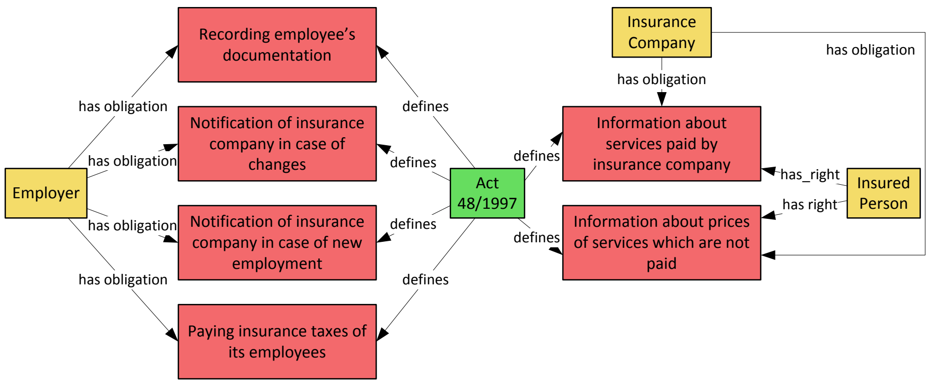
Recording employee's documentation
Notification of insurance company in case of changes
Notification of insurance company in case of new employment
Paying insurance taxes of its employees
Employer
has obligation
has obligation
has obligation
has obligation
defines
defines
defines
defines
Act 48/1997
defines
defines
Insurance Company
has obligation
has obligation
Information about services paid by insurance company
Information about prices of services which are not paid
has_right
has right
Insured Person

Fig. 1. Sample of semantic concepts extracted from Public Health Act valid in Czech Republic

3 Current Czech Legislation and Related Systems

In the Czech Republic, there currently exist several systems that provide an access to (a subset of) law, court decisions or other related information in an electronic form. Some of them even claim to provide the *consolidated versions of acts*, nevertheless none of them is an official version to be approved by the Head of the Parliament (who is responsible for it). If fact, even the members of the Czech Parliament work with these systems, i.e. with unofficial data. The solution to the problem is being provided in two closely related projects proposed by the Czech Government – *eSbirka*[3] and *eLegislativa*[4] – whose aim is (1) to provide official and approved version of the consolidated versions of acts in the electronic form and available to anyone and (2) to enable to speed up the legislature process in the Czech Republic via direct amending of these official electronic acts. The problem of these systems is currently the financial support. It is not the question of preparing an electronic version of the documents or suitable interfaces for various types of users (a citizen, a member of the Parliament, a Head of the Parliament, etc.), but it requires a tremendous effort of experts in law to solve known ambiguities, to study historical acts that are still valid and not in accordance with newer acts, etc.

In the following sections we provide a brief overview of existing systems that enable to browse and query (a part of) the Czech legislation.

3.1 ASPI

System *ASPI*[5] from the Wolters Kluwer, Czech Republic is currently one of the most popular systems that enable to browse and query electronic version of legislation and related data. In addition, being a publishing company, the vendor provides an interesting and important extension – an access to related basic *literature* where various acts are

[3] eLegislation in English
[4] eLegislature in English
[5] `http://www.systemaspi.cz/` [in Czech]

explained, commented and discussed. Considering the browsing and querying aspects, it supports full-text search supporting also all grammatical forms of Czech or Slovakian respectively. The search can cover all texts, or selected parts such as titles, content, appendices, notes, or tables of contents. The system enables to filter the documents according to meta data, such as identification (e.g. file numbers of court decisions), date of issue (valid versions or older versions), or issuing institution (e.g. the Constitutional Court, the Supreme Administrative Court etc.).

3.2 LexGalaxy

LexGalaxy[6] is another tool which enables to browse and search the legislation. It involves a selection of 100,000 documents from the constitutional order of the Czech Republic from 1918. The system includes also a selected information on law of the European Union and the European Court of Human Rights. The search in the legislation can be done in three ways – using full-text search, document identifications, or indexes. Full-text search enables to specify the searched area (e.g. paragraph, title, appendix, etc.) and supports various grammatical forms of the searched words. The indexes involve types of documents (e.g. acts, Constitutional Court decisions, etc.), date of issue, validity, issuing institution, etc. The search conditions can be combined using logical and proximity operators.

3.3 Public Administration Portal *Portal.Gov.cz*

The Public Administration Portal *Portal.Gov.cz*[7] involves various information for citizens, entrepreneurs and businessmen, foreigners living in the Czech Republic, and public authorities. The functionalities involve also a module for simple full-text search in legislation. It enables to search in texts of acts, their titles or according to their number.

4 Current Research Projects and Foreign Solutions

The first research solutions in the considered area are the techniques for automated *categorization* of documents or their parts based on *machine learning* [17]. They are able to assign each document or its part (e.g. a paragraph) a category from a given set. The methods achieve good results in case of categorization of collections of documents from a narrowly focused domain of interest.

The next step in this field is the usage of an *ontology* instead of a set of categories. An ontology formally describes the semantics of the domain of interest; however, in addition to the categories of objects they also describe possible relationships between objects. The aim is to interpret individual parts of the document just against the ontology which is actually an extension to the methods described in [17]. The current literature is currently focussed in extending these applications in the biomedical field [4]. In the area of legislative data; however, their application has not yet been sufficiently explored.

[6] `http://www.legsys.cz/` [in Czech]
[7] `http://portal.gov.cz/app/zakony/?path=/portal/obcan/` [in Czech]

The interpreted objects and relationships between them can be further effectively represented in the RDF data model. At present there are many methods for database processing of RDF data, although yet especially at the level of basic research [10].

4.1 The JURIX Conference

The *Foundation for Legal Knowledge Based Systems* (JURIX)[8] is a forum for researchers in the field of Law and Computer Science in the Netherlands and Flanders. From the point of view of INTLIB we can find several interesting papers which deal with enhancing the way legislation is searched using semantics of the data.

Paper [14] distinguishes the relevant approaches into *knowledge-engineering*, involving artificial intelligence or case-base reasoning, and *natural language processing*. The authors claim that the former class suffers from several problems, such as domain specificity or high financial cost, and they argue that natural language processing is promising. Paper [12], similarly, analyzes whether machine learning techniques or knowledge-engineering approaches are better for classification of sentences in laws. The conclusion is that both the approaches reach similar results; however, the machine learning techniques are naturally sensitive to the training set and its correspondence to the analyzed data. In paper [13] the authors apply a thesaurus-based statistical indexing technique to retrieve relevant case law from 68,000 court verdicts. It is based on classical vector space model extended with thesaurus so that only terms from a particular domain are considered. Finally, paper [8] describes the results of a study consisting of two tasks: (i) how the "obligation" Fundamental Legal Concept is differently represented in the FrameNet[9] resource, in terms of Semantic Frames, and (ii) how the concept of "public function" stemmed from the "obligation" Fundamental Legal Concept can be ontologically characterized. The *FrameNet* project is building a lexical database of English that is both human- and machine-readable, based on annotating examples of how words are used in texts.

In general, the papers prove that our aim is right and that it is crucial to create a general system that enables to work with the legislation data in a more sophisticated way provided by extending them with semantics. The experiments show that the strategy is promising; however, such a system is still missing.

4.2 Plans of the European Union

Another important related work can be found in strategies and plans of the European Union. The final report of Working group "Indexing and Search" [7] identifies and recommends best practices and technologies which highly correlate with our aim and which thus confirms that the strategy is promising a should be further extended.

5 INTLIB Architecture

As we have mentioned in the Introduction, there are two key parts of the INTLIB project – extraction and presentation, i.e. creating the knowledge base and its user-friendly

[8] `http://www.jurix.nl/`

[9] `https://framenet.icsi.berkeley.edu/fndrupal/`

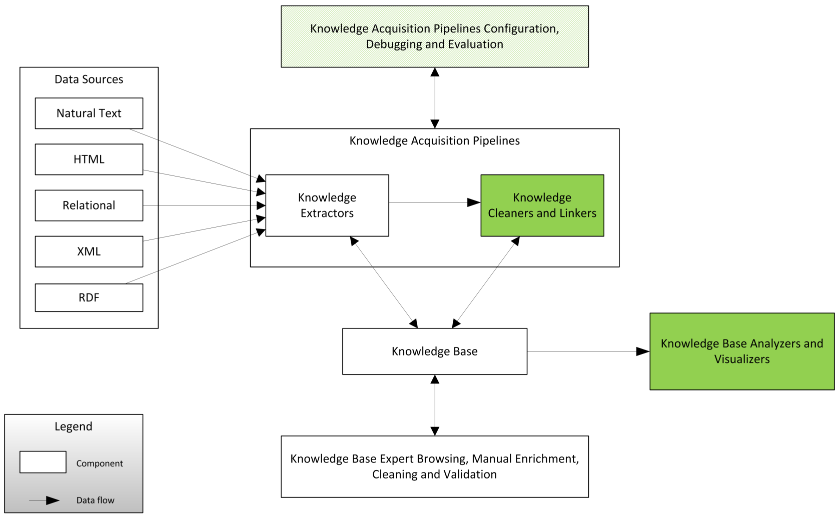
Knowledge Acquisition Pipelines Configuration, Debugging and Evaluation
Data Sources
Natural Text
HTML
Relational
XML
RDF
Knowledge Acquisition Pipelines
Knowledge Extractors
Knowledge Cleaners and Linkers
Knowledge Base
Knowledge Base Analyzers and Visualizers
Knowledge Base Expert Browsing, Manual Enrichment, Cleaning and Validation
Legend
Component
Data flow

Fig. 2. Architecture of system INTLIB

browsing. Since our aim is a general framework utilizable and extensible for particular problem domains with plug-ins, the architecture is more complex. It is based on the idea of *pipelines* which specify the selected steps of the process.

5.1 System Architecture

The architecture of the system is depicted in Figure 2. On the input we can assume various types of data, i.e. not only documents with a natural text in, e.g., PDF [3] format (i.e. in our case acts and court decisions), but also HTML [16] or XML [11] documents, data stored in a relational database, RDF [9] triples etc.

The data are first provided to the *knowledge acquisition pipelines* which extract the knowledge base, i.e. the objects described in the data, their relations and properties. The pipelines need to be first configured, i.e. particular modules need to be selected. The configured pipeline also needs to be debugged and evaluated. The pipelines consist of *knowledge extractors* and *knowledge cleaners and linkers*. The cleaners ensure, that the data extracted from different data sources of various quality are cleaned, e.g., the duplicities and false candidates are removed. The linkers map the newly extracted data to the current data in the knowledge base.

Apart from automatic knowledge base extraction, cleaning and linking, the system also involves a user interface that enables browsing the knowledge base and its manual enrichment (i.e adding new components), as well as cleaning and linking. The aim of this part is:

1. to provide a preliminary interface which enables to create at least a basic knowledge base for testing related modules,

2. to enable to add data that cannot be added manually (due to various reasons, such as, e.g., confidentiality of data sources or limitations of the automatic knowledge base extraction part), and
3. to enable to confirm or refute candidates for linking, discarding, unifying etc.

An emphasis is put on the GUI, because in this case we assume a user which is an expert in the particular domain (i.e. a lawyer), but not in the RDF representation and related technologies such as SPARQL. In preliminary stages of the implementation the module provides all possible candidates to be confirmed/refuted. Later we will add also filtering and cleaning modules that show only possible candidates for correction of discarding.

The last but not least part of the system involves the module which enables to *visualize the knowledge base* and in particular *analyze its content* in a user friendly manner. The analytical part involves an advanced query interface including smart hints, learning from previous user queries and results, etc. For the purpose of visualization of the data we will utilize the SW project *Payola* [5] which enables to visualize graph data in a user friendly manner.

5.2 System Pipelines

Systems pipelines can in general form an acyclic graph whose nodes represent particular *modules* which process the data and edges represent inputs and outputs of the modules. In general, the exchanged data can be of any format that is understood by the respective output/input module; however, for our case and for the sake of simplicity and clarity we assume RDF data in all the cases (if not stated otherwise).

The idea of pipelines results from a SW project *ODCleanStore* [6] which supports textual configuration of pipelines and respective modules. In case of INTLIB the user interface is more friendly, providing a graph visualization of the pipeline and a set of forms that enable to fill in the respective parameters of the particular modules (based on the idea of *XForms* [2] and *VAADIN* [1]).

Examples of two use cases represented by pipelines are depicted in Figure 3. In the former case we can see a set of pipelines for extraction of references among acts, in the latter case a set of pipelines which recognizes structure and terms used in particular acts. In both the cases first the *annotation pipeline* annotates “interesting” parts of the input data, i.e. substrings of acts that represent references, terms, parts etc. Next, the *extraction pipeline* processes the annotated text and select parts which truly represent particular items. The *transformation pipeline* deals with cleaning of the extracted data, whereas it can be even empty when we know that the previous steps cannot produce duplicities. Finally, the *loader* ensures loading of the extracted and cleaned data into the knowledge base, including the linking phase.

The system involves also a *scheduler* which enables to run the pipelines periodically (e.g. in case a pipeline crawles data refreshed or extended gradually), after another pipeline finishes, etc. Similarly, for the purpose of debugging the system involves also a *debugger*. It enables, e.g., to log intermediate results of particular modules of pipelines, run the modules in a debugging mode with extended reports on status, or running only a part do the pipeline (e.g. a selected path or subgraph).

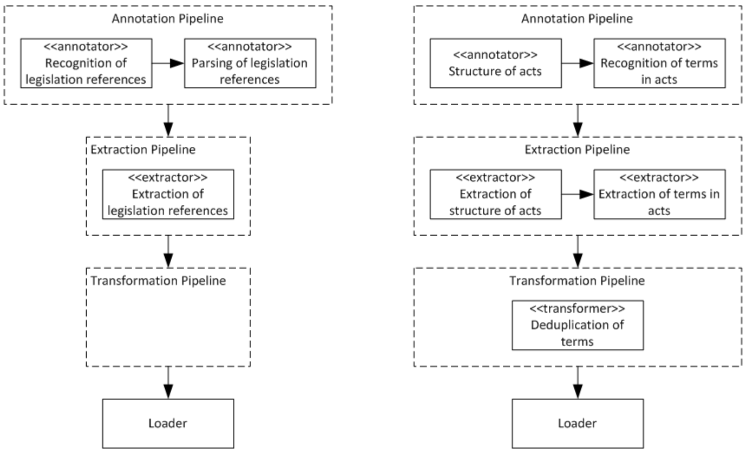
Annotation Pipeline
<<annotator>> Recognition of legislation references
<<annotator>> Parsing of legislation references
Extraction Pipeline
<<extractor>> Extraction of legislation references
Transformation Pipeline
Loader
Annotation Pipeline
<<annotator>> Structure of acts
<<annotator>> Recognition of terms in acts
Extraction Pipeline
<<extractor>> Extraction of structure of acts
<<extractor>> Extraction of terms in acts
Transformation Pipeline
<<transformer>> Deduplication of terms
Loader

Fig. 3. Sample pipelines of the system

5.3 LEX Ontology

The goal of the LEX ontology is to enable to represent the legislation (sources of law) in a machine-readable form conforming to Linked Data principles. Data from other data sources can be linked to legislation represented according to the LEX ontology and, therefore, enriched with the legislative information for further processing by machines. Legislation can also be linked to other data sources.

As we have already indicated, there are different kinds of sources of law:

- *act* is a source of law enacted by a national or regional parliament,
- *decree* is a source of law issued by a national or regional government, ministry, or another public authority
- *regulation* is a source of law issued by a national or regional government, ministry, or another public authority which complements and/or specifies an act,
- *court decision* is a source of law issued by a court as an official decision in a particular legal case,

A source of law can also change another source of law. In that case, we call the source of law *amendment*. An amendment of a given kind can change a source of law which is of the same kind. For example, an act may change another act.

The LEX ontology introduces the following classes for the kinds of sources of law mentioned above: `lex:SourceOfLaw` for sources of law of all kinds (superclass of all other classes), `lex:Act` for acts, `lex:Decree` for decrees, `lex:Regulation` for regulations, and `lex:Decision` for court decisions. (We omit the respective UML diagram for simplicity and space limitations.)

Sources of law of most kinds (except of court decisions) exist in different versions. Some versions are outdated, at most one version is currently valid, and some versions are enacted but have not come to force yet. From this viewpoint, it is reasonable to represent a source of law as an abstract notion of intellectual creation which is independent of

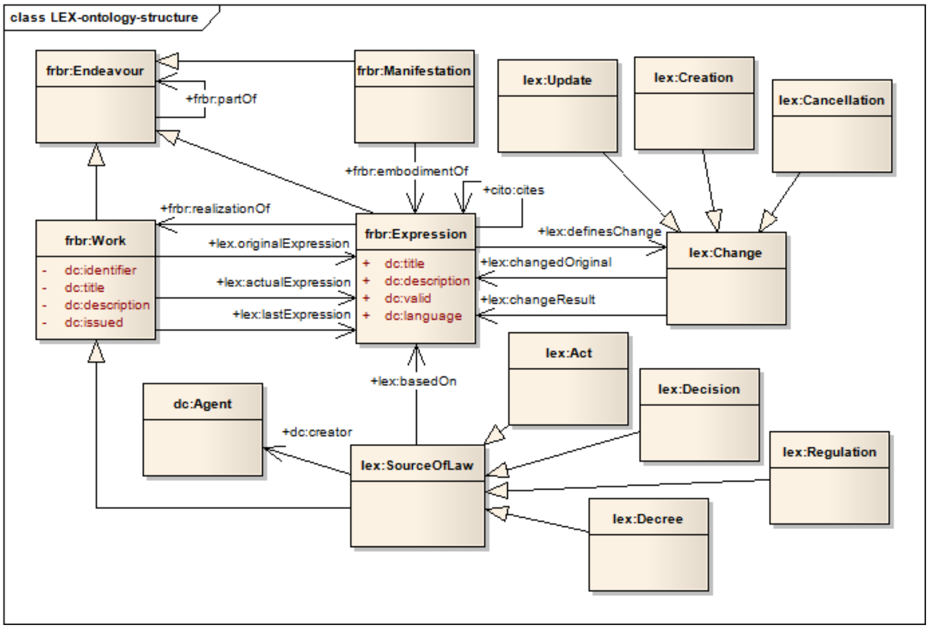

Fig. 4. Legislation Ontology LEX

particular versions of the source. Moreover, each version of the source as well as its each physical embodiment should have representation on its own. This logic is built into the LEX ontology. However, we do not introduce own ontological constructs but reuse the Functional Requirements for Bibliographic Records (FRBR)[10] ontology (as depicted in Figure 4). We reuse the following three FRBR classes: `frbr:Work` for abstract notions of an intellectual creation which are sources of law, `frbr:Expression` for particular versions of sources of law, and `frbr:Manifestation` for particular documents which are physical embodiments of particular versions of sources of law. The usage of FRBR allows us to distinguish a source of law itself, its particular versions and their physical embodiments. From the linked data point of view, it is therefore possible to link and query the source of law as an abstract entity which is independent of particular versions of the source. It is also possible to link and query its particular versions and also their amendments.

We also reuse two FRBR properties: `frbr:realizationOf` to link a version (member of `frbr:Expression`) to its source of law (member of `frbr:Work`) and `frbr:embodimentOf` to link a document (member of `frbr:Manifestation`) to a version of a source of law it is embodiment of (member of `frbr:Expression`). Because each source of law (member of `lex:SourceOfLaw`) is also a member of `frbr:Work` we set `lex:SourceOfLaw` as a subclass of `frbr:Work`.

For a given source of law we need to know its currently valid version, original version (i.e. the first version), and the last enacted version (which have not necessarily

[10] `http://www.loc.gov/cds/downloads/FRBR.PDF`

needed to come to force yet). For this, we introduce three new properties in the LEX ontology: `lex:originalExpression` to link the original (first) version to the respective source of law, `lex:actualExpression` to link the currently valid version to the respective source of law, and `lex:lastExpression` to link the last enacted version to the respective source of law.

Last but not least, we also model changes in legislation with `lex:Change` class. We distinguish three subclasses for three specific kinds of changes: `lex:Creation` to model that something new has been created, `lex:Cancellation` to model that something existing has been removed and `lex:Update` to express that something existing has been updated. More information on the LEX ontology can be found in [15].

6 Conclusion

The aim of this paper was to describe the first phase of project *INTLIB – an INTelligent LIBrary*, i.e. analysis of the problem domain, architecture of the project and related ontology for legislation documents. The target of the project is to provide a general framework for extraction of knowledge from input data (of any kind) so that more advanced querying than usual full-text is possible. Using plug-ins it can be utilized for a particular kind of data – in the first phase the legislation documents.

In the following phases of the project we will naturally focus on implementation of all parts of the system described in Section 5. The key emphasis will be first put on the user interface, configuration and evaluation parts and interfaces between the modules, so that in the next phase we can focus on implementation of all the related plug-ins for legislation processing. As a future work we plan to use the system in other applications, such as environmental reports, policies of companies, etc.

References

1. *VAADIN.* W3C Recommendation, 20 October 2009. `https://vaadin.com/`.
2. *XForms 1.1.* W3C Recommendation, 20 October 2009. `http://www.w3.org/TR/xforms/`.
3. *ISO 32000-1:2008: Document Management – Portable Document Format.* Adobe, 2008.
4. Current Issues in Biomedical Text Mining and Natural Language Processing. *Journal of Biomedical Informatics*, 42(5):757 – 759, 2009.
5. *Payola.* Student SW Project, Charles University in Prague, Czech Republic, 2012. `https://github.com/payola/`.
6. *ODCleanStore.* Student SW Project, Charles University in Prague, Czech Republic, 2013. `http://sourceforge.net/p/odcleanstore/home/Home/`.
7. A. Gola et al. *Working Group Indexing and Search – Final Report.* European Forum of Official Gazettes, Riga, Latvia, 2011.
8. T. Agnoloni, M. Fernandez, M. Sagri, D. Tiscorni, and G. Venturi. When a FrameNet-Style Knowledge Description Meets an Ontological Characterization of Fundamental Legal Concepts. In *AI Approaches to the Complexity of Legal Systems*, volume 6237 of *LNCS*, pages 93–112. Springer Berlin Heidelberg, 2010.

9. D. Beckett. *RDF/XML Syntax Specification (Revised).* W3C, February 2004. `http://www.w3.org/TR/rdf-syntax-grammar/`.
10. C. Bizer, T. Heath, and T. Berners-Lee. Linked Data – The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.
11. T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau. *Extensible Markup Language (XML) 1.0 (Fifth Edition).* W3C, 2008. `http://www.w3.org/TR/xml/`.
12. E. de Maat, K. Krabben, and R. Winkels. Machine Learning versus Knowledge Based Classification of Legal Texts. In *JURIX 2010*, pages 87–96, Amsterdam, The Netherlands, 2010. IOS Press.
13. M. Klein, W. van Steenbergen, E.Uijttenbroek, Arno Lodder, and F. van Harmelen. Thesaurus-based Retrieval of Case Law. In *JURIX 2006*. IOS Press.
14. K. T. Maxwell and B. Schafer. Concept and Context in Legal Information Retrieval. In *JURIX 2008*, pages 63–72, Amsterdam, The Netherlands, 2008. IOS Press.
15. M. Necasky, T. Knap, J. Klimek, I. Holubova, and Barbora Vidova-Hladka. Linked Open Data for Legislative Domain – Ontology and Experimental Data. In *LIT 2013, Poznan, Poland*, pages 172 – 183, Poznan, Poland, 2013. Springer-Verlag.
16. D. Raggett, A. Le Hors, and I. Jacobs. *HTML 4.01 Specification.* W3C, December 1999. `http://www.w3.org/TR/html401/`.
17. F. Sebastiani. Machine Learning in Automated Text Categorization. *ACM Comput. Surv.*, 34(1):1–47, March 2002.

Spatial Clustering of Disease Events Using Bayesian Methods

Lukáš Marek, Vít Pászto, Pavel Tuček, and Jiří Dvorský

Department of Geoinformatics, Faculty of Science, Palacky University in Olomouc
17. listopadu 50, Olomouc, 771 46, Czech Republic
{lukas.marek, pavel.tucek}@upol.cz, vit.paszto@gmail.com,
jiri.dvorsky@vsb.cz
http://www.geoinformatics.upol.cz/

Abstract. One of main aims of the spatial analysis of health and medical datasets is to provide additional information to the specialized medical research. These analyses can be used for disease mapping; searching for places with a higher intensity and probability of the disease event; or the influence assessment of selected natural or artificial phenomena. Suitably selected methods allow a proper analysis of these data and identification of irregularities and deviations of the phenomena in the area of interest. The structure of medical data usually needs to be standardized (over age structure of the population) before the comparison of different regions. Bayesian statistics derives the posterior probability as a consequence of a prior probability and a probability model for the data observed. Geosciences and geomedicine usually use the Bayesian theory for smoothing of data - to help depict the real spatial pattern and its changeability. The Bayesian principles, together with the spatial neighbourhood and statistical models, are successfully used also for the identification of spatial and space-time clusters with significantly higher/lower risk of incidence of the disease. These procedures are denoted as methods of spatial clustering and can be used with or without utilization of properties of certain phenomena. Particularly, occurrence data of campylobacteriosis infection in four Moravian regions in period 2008 – 2012, which were provided by The National Institute of Public Health, were used for the case study.

1 Introduction

The disease mapping, visualization of disease rates and the clustering of disease data are still one of the most interesting topics in geosciences. It is because of the nature of the data which are often pure spatial with rich descriptive part and it is easy to combine them with other data (demographic, economic, etc.) [14]. This contribution aims to present the usage of empirical Bayesian methods in the disease mapping and subsequent creating of disease maps. Bayesian methods incorporate the prior knowledge about the phenomenon (or underlying processes) to provide more accurate and easily understandable description of the situation. Empirical Bayesian procedures are used for disease rates smoothing in the case of choropleth map. They also help to identify local clusters of more/less affected areas. The main topic of the case study in this paper is the analysis of the spatial distribution of disease called campylobacteriosis in

Moravian regions between years 2008 and 2012 with usage of Bayesian estimates based on Poisson distribution.

2 Case study and Data

The case study, where further described methods are applied, is dealing with the spatial distribution of the campylobacteriosis in four Moravian regions (Moraskoslezsky, Zlinsky, Olomoucky and Jihomoravsky) between years 2008 and 2012. There were almost 49 thousand of cases of the disease during that period, while only 34 thousand were expected according to previous records. Using disease counts and disease rates calculated for the municipalities in the area of interest, we tried to identify areas that are possibly more vulnerable to the disease than their neighbourhood. The 5-year observed number of cases, expected number of cases and relative risk (SIR) were used as main disease characteristics for this study.

Campylobacteriosis is caused by bacteria called Campylobacter jejuni, which is found worldwide in the intestinal tracts of animals. The bacteria are spiral shaped and can cause disease in animals and humans. Most cases of campylobacteriosis are associated with handling or eating raw or undercooked poultry meat or fresh milk. Campylobacteriosis causes gastrointestinal symptoms, such as diarrhoea, cramping, abdominal pain, and fever in domestic animals and humans. Young animals and humans are the most severely affected [23].

2.1 Data

The data set for this study was provided by The National Institute of Public Health of the Czech Republic. The database contains almost 50 thousands records of the campylobacteriosis occurrence in the period 2008 – 2012. Names, surnames, identity numbers and sometimes also the full addresses are not included because it is treated with sensitive personal data. The data were firstly cleansed of inconsistencies and then the geocoding process was run. Furthermore, the individual records were aggregated to the municipalities - administrative units - due to the clarity of the visualization and analyses [15]. The problem of the conversion of spatial phenomena between different areal or administrative units is well known as MAUP – Modifiable Area Unit Problem [18]. During the calculation of disease rates and expected number of cases, the population data from the Population and Housing Census of the Czech Republic were used as the main basis for the data standardization.

Figure 1 shows the probability density function of disease events counts, total population and standardized incidence ratio in Moravian municipalities visualized in the logarithmic scale (upper graph) and in the logarithmic scale and centred (lower graph) in order to simplify visual analysis. The probability function of population and diseases events counts are fairly similar, which indicates the need for standardization and also analysis that considers this close relation. Figure 2 then depicts the spatial distribution of standardized incidence ratio (SIR). SIR is the ratio of the number of disease cases observed in the study group or population to the number that would be expected if the study population had the same specific rates as the standard population, multiplied by 100 and usually expressed as a percentage [10]. By this way, SIR expresses relative risk (or vulnerability) of the municipality to certain disease. Municipalities traversing value of 1 are more vulnerable to disease, while municipalities with SIR lower than 1 are healthier.

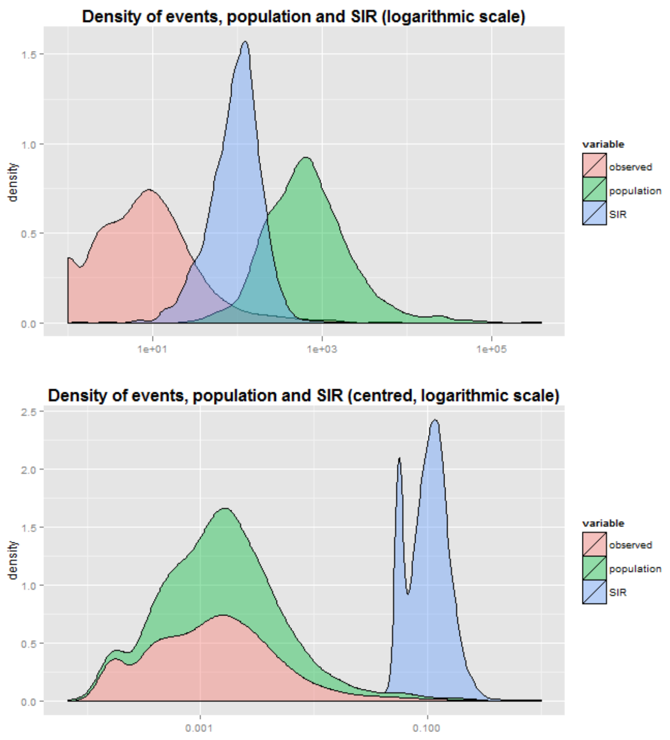

Fig. 1. The probability density function of disease events counts (red), total population (green) and standardized incidence ratio (blue) in Moravian municipalities visualized in the logarithmic scale (upper graph) and in the logarithmic scale and centred (lower graph) in order to simplify visual analysis.

3 Methods

During the study of disease spatial distribution, mainly in the case of aggregated data, it is often suitable to focus on the local variability of the disease occurrence or relative risk rather than examine the study area as a whole. This procedure is usually denoted the disease cluster detection. The general review of methodology as well as usage of spatial clustering methods and its Bayesian enhancements in the literature, e.g [6, 11, 21] etc.

In geosciences the spatial clustering is often encapsulated as the analysis of the spatial autocorrelation. The spatial autocorrelation is the correlation among values of a single variable, which is strictly attributable to their relatively close locations on a two-dimensional (2-D) surface, introducing a deviation from the independent observations assumption of classical statistics [7]. Positive spatial autocorrelation refers to the patterns where nearby or neighbouring values are more alike; while negative spatial autocorrelation refers to the patterns where nearby or neighbouring values are dissimilar. One can distinguish two main types of spatial autocorrelation, which are global and local. The null hypothesis for global clustering is simply that no clustering exists (i.e. random spatial dispersion ≈ CSR). Probably the most used method for both global and local analyses of spatial autocorrelation is Moran's I statistics (together with e.g. Getis-Ord G and Geary's C statistics). Moran's I coefficient of autocorrelation is similar to Pearson's correlation coefficient, and quantifies the similarity of an outcome variable among areas that are defined as spatially related [16]. The problem with variance instability for rates or proportions, which served as the motivation for applying smoothing techniques to maps may also affect the inference for Moran's I test for spatial autocorrelation [1]. The implementation of the adjustment procedure of Assuncao and Reis (1999), which uses a variable transformation based on the Empirical Bayes principle may be one of solutions. This yields a new variable that has been adjusted for the potentially biasing effects of variance instability due to differences in the size of the underlying population at risk [1].

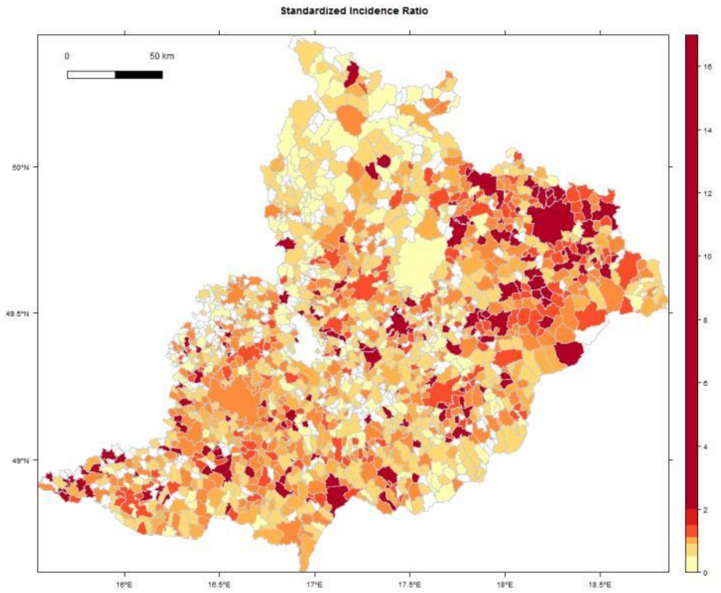

Fig. 2. Choropleth map of standardized incidence ratio, which is generally the ratio between observed disease cases and its potential (expected) amount, which is based on the population and its age structure in individual municipalities.

3.1 Spatial clustering of case events data

If a cluster is described as an uncommon collection of events, then it is needed to detect these collections observed within the data set. Such methods define a set of potential clusters, collections of events each of which we might define as a cluster if the collection appears unusual enough (discrepant from the null model of interest), then identifies the most unusual of these [21]. This general idea motivated the "geographical analysis machine" (GAM) of Openshaw where potential clusters were defined as collections of events falling within circular buffers of varying radii [17]. The buffers were centred at each point in a fine grid covering the study area and the GAM approach mapped any circle whose collection of events were detected as unusual, e.g., those circles where the number of events exceeded the 99.8th percentile of a Poisson distribution with mean defined by the population size within the buffer multiplied by the overall disease risk [21]. GAM is very useful for descriptive purposes, but should not be used for hypothesis testing.

Scan statistics provide another approach that is similar to the local case/control ratios. A scan statistic involves definition of a moving window and a statistical comparison of a measurement (e.g., a count or a rate) within the window to the same sort of measurement outside the window. Kulldorff [8] defines a spatial scan statistic very similar to the GAM and other methods, but with a slightly different inferential framework. The primary goal of a scan statistic is to find the collection(s) of cases least consistent with the null hypothesis, i.e. the most likely cluster(s) but Kulldorff goes a bit further and seeks to provide a significance value representing the detected cluster's unusualness, with an adjustment for multiple testing [22]. Kulldorff [8] considers circular windows with variable radii ranging from the smallest observed distance between a pair of cases to a user-defined upper bound. He builds an inferential structure based on earlier works where authors note that variable-width one-dimensional scan statistics represent collections of local likelihood ratio tests comparing a null hypothesis of the constant risk hypothesis compared to alternatives where the disease rate within the scanning window is greater than that outside the window. The maximum observed likelihood ratio statistic provides a test of overall general clustering and an indication of the most likely cluster(s), with significance determined by Monte Carlo testing of the constant risk hypothesis [22].

The outstanding description of methods including their mathematical apparatus or their possible implementations and applications provide mainly [8, 9, 17].

3.2 Bayesian mapping and spatial clustering of case events data

Presentation of disease rates in area units as choropleth maps can inadvertently provide misleading information. This fact is well known mainly in the case of small-area studies that introduces an extra source of variability into the map because of random variation. Typically, sparsely populated areas with few (or zero) cases can generate extreme values of the SMR (and also prevalence), as the variance of the SMR is inversely related to expected number of cases and small populations have large variability in the estimated rates [5] and that is why risk estimates and other rates are rather unstable.

Bayesian methods provide a solution for this kind of bias. They use probability models to obtain smoothed estimates consisting of a compromise between the observed rate for each region and an estimate from a larger collection of cases and persons at risk (e.g., the rate observed over the entire study area or over a collection of neighbouring regions) [22]. The basic principle of Bayesian methods is that uncertain data can be strengthened by combining them with prior information [19]. In the case of empirical Bayes estimation of spatially-varying disease risk, posterior risk can be estimated from a weighted combination of the local risk (also called the likelihood) and the risk in surrounding areas, the latter representing the prior information [4].

The set of areal units on which data are recorded can form a regular lattice or differ largely in both shape and size, so data typically exhibit spatial autocorrelation, with observations from areal units close together tending to have similar values. A proportion of this spatial autocorrelation may be modelled by including known covariate risk factors in a regression model, the residual spatial autocorrelation can be induced by a number of factors, and violates the assumption of independence that is common in many regression models [12]. The most common remedy for this residual autocorrelation is to augment the linear predictor with a set of spatially correlated random effects, as part of a Bayesian hierarchical model. These random effects are typically represented with a conditional autoregressive (CAR) model, which induces spatial autocorrelation through the adjacency structure of the areal units. However, the CAR priors force the random effects to exhibit a single global level of spatial autocorrelation, ranging from independence through to strong spatial smoothing. Such a uniform level of spatial smoothness for the entire region is unrealistic for real data, which are instead likely to exhibit sub-areas of spatial autocorrelation separated by discontinuities. Such localized spatial smoothing may occur where rich and poor communities live side-by-side, and in this context the response variable is likely to evolve smoothly within each community with a sudden change in its value at the border where the two communities meet [12].

To be more particular, the analysis provided in the case study is based on the function that fits a Poisson log-normal random effects models to spatial count data, where the random effects are modelled by the localised conditional autoregressive (CAR) model proposed by [13]. The random effects in neighbouring areas (e.g. those that share a common border) are modelled as correlated or conditionally independent, depending on whether the populations living in the two areas are similar (correlated random effects) or very different (conditionally independent). The model represents the natural log of the mean function for the set of Poisson responses by a combination of covariates and a set of random effects. Inference is based on Markov Chain Monte Carlo (MCMC) simulation, using a combination of Gibbs sampling and Metropolis steps [12]. The outstanding overview of Bayesian techniques are provided in [11, 12] and others.

4 Results

Firstly, the original data of disease events needed to be aggregated to the municipality level, filtered to selected area of the Czech Republic. Subsequently, aggregated counts that represented actually observed cases served as the bases for the calculation of expected number of cases in the area that were found out using internally indirect standardization. SIR, which is the ratio between observed and expected number of cases and expresses the relative risk of the area can be seen in the Fig. 2.

Both values served as inputs for the Openshaw's Geographical Analysis Machine that allowed the identification of possible disease clusters in the area. Radius for the analysis was chosen as 7 km, the alpha value for the cluster identification was 99.8

quantile of the Poisson probability distribution. Several significant clusters can be identified throughout the study area (Fig. 3), but 2 most visible can be seen – the first is located in the southern part of the area near the Brno municipality, while the second cluster is placed in densely populated surroundings of Ostrava (but surprisingly except the city itself). As it was mentioned before, GAM is very useful for descriptive purposes, but should not be used for hypothesis testing because of the overestimation of clusters. That is why other methods - scan statistics and Bayesian identification of inference in the area, were performed.

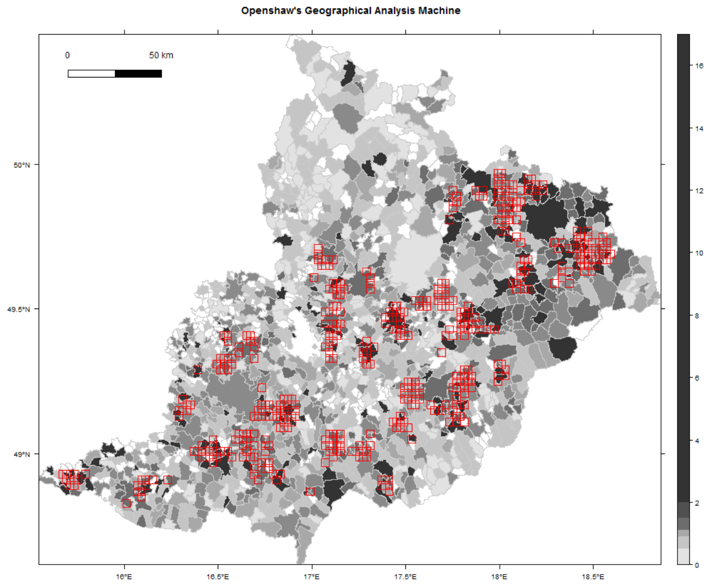

Fig. 3. Identification of spatial disease clusters of campylobacteriosis with the use of Openshaw's Geographical Analysis Machine. Colours and legend depicts standardized incidence ration, while red squares identify locations that are involved in probable disease spatial clusters.

Results of scan statistics used for the identification of spatial disease clusters of campylobacteriosis with the use of the clustering function for Kulldorff and Nagarwalla's statistic are shown on the Fig. 4. The scan statistics is based on the Poisson distribution of disease events, 15 % significance and 5 % fraction of total population. Unlike GAM results, only one significant cluster was identified in the northern part of the study area and it is located in the surrounding of the Ostrava with the core in the village Kateřinice (dark grey area on the Fig. 4).

The last analysis is based on the function that fits a Poisson log-normal random effects models to spatial count data, where the random effects are modelled by the localised conditional autoregressive (CAR) model. The model is based on the list of binary neighbourhood with the queen contiguity conceptualization of space. The observed amount of cases is modelled as the of logarithmical scale of amount of expected number of disease events (intercept) and the ratio between young people (under 15) and elderly people (64+), which is also the basis of the dissimilarity matrix. The analysis detected only two areas (Fig. 5 - left part) that might be the cores of possible clusters. The first municipality is located in the south-western part of the study area (village Podhradí nad Dyjí). The second theoretical core area is placed in the village Nelepeč-Žernůvka in the west of the study area. That might indicate other necessary customization of the model with the use of other characteristics of the area. Similar analysis based on the distribution of the population was performed due to the comparison. It is depicted in the right part of Fig. 5. Unlike the previous analysis, the result showed significantly more borders between clustering areas and their neighbourhood. On the other hand most of them are densely populated, so the analyst should consider their importance carefully and focus on several individual locations.

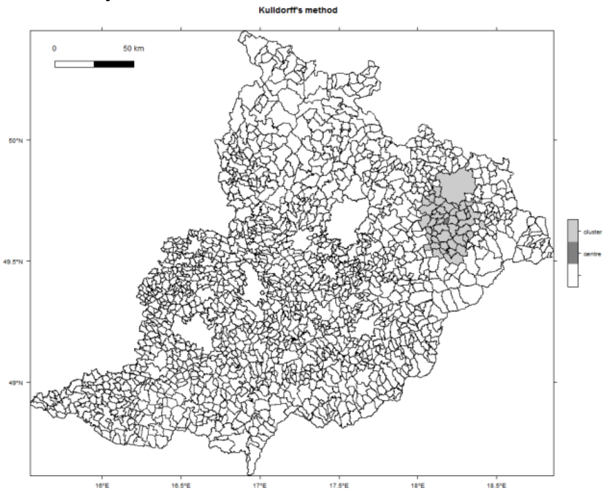

Fig. 4. Identification of spatial disease clusters of campylobacteriosis with the use of the clustering function for Kulldorff and Nagarwalla's statistic. Dark grey areas stand for central (core) area, light grey colour stand for other municipalities in the spatial cluster.

5 Discussion and Conclusion

The contribution aimed to introduce methods of spatial clustering and Bayesian spatial clustering that were based either on the location of disease events in the study area of four Moravian regions or their locations and demographical characteristics of municipalities. One has to realize that all presented methods are dependent on the scale and also on the prior information, which is entering the models mostly in the form of the probability distribution. Therefore, results and their evaluation have to be performed carefully in order to avoid misinterpretation. The aim of the contribution is therefore not only to use methods in real case study but also to show several different results that originally come from the same data.

Firstly Openshaw's GAM detected high number of possible diseases clusters, but due to its disadvantages, results were taken just as informative and an initial step for further analysis. Then, scan statistics based on Kulldorff and Nagarwalla's statistic was used for the identification of spatial disease clusters of campylobacteriosis. The scan statistics discovered one statistically significant cluster on the north of the study area. Lastly, the Poisson log-normal random effects models to spatial count data, where the random effects are modelled by the localised conditional autoregressive (CAR) model, was used to proceed more detailed and complex analysis. This model incorporated the information about neighbourhood of individual municipalities and also the dissimilarity matrix based on the age structure of the population in the neighbouring villages or cities. The model was able to identify two core areas of possible clusters.

One has to realize that Bayesian techniques usually tend to shift values to the mean risk – global or local by incorporating information between areas. The risks in areas with more information (e.g., urban areas) are usually less smoothed than in areas that exhibit higher sampling variation (typically those with low number of cases), and thus produce more stable estimates of the pattern of underlying disease risk [20]. However, although raw risks can produce "noisy" maps that are difficult to interpret, over-smoothed maps may produce a homogeneous risk surface, masking the true risk distribution [3]. It is important to mention that all analyses presented in this paper are heavily dependent on the scale. We chose the scale of municipal districts but results on other scales could show differences. When someone chooses to broad scale for the analysis, results will probably reveal one (or several) large cluster so the local variance disappears. On the other hand, to local scale may not lead to identification of any clusters. The extension of Bayesian model using other characteristic of the population, spatial unit or disease is possible; however their dynamic properties are mainly shrunk to the sequential procession of time series or time slices.

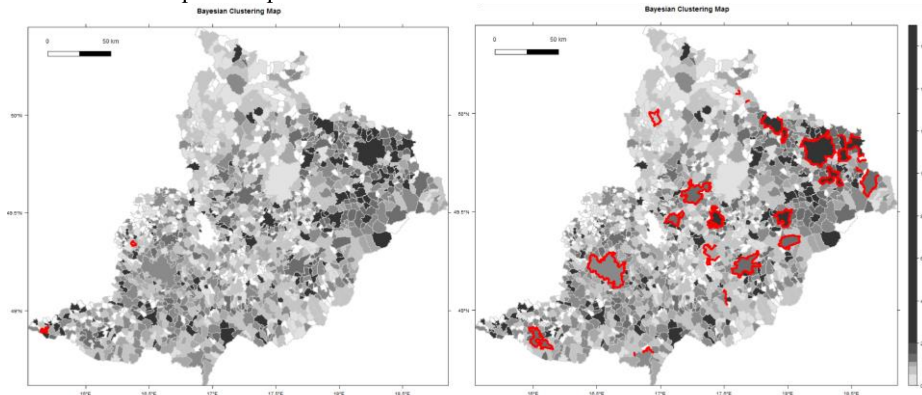

Fig. 5. Identification of spatial disease clusters of campylobacteriosis with the use of localised conditional autoregressive (CAR) model based on dissimilarity metrics with binary neighbourhood relations to spatial Poisson data. Colours and legend depicts standardized incidence ration, while red areas identify locations that are centres of probable disease spatial clusters. The left part describes the relation of likely clusters to the ratio of old people to children, while the right part is based on the population.

Acknowledgement

The authors gratefully acknowledge the support by the Operational Program Education for Competitiveness - European Social Fund (project CZ.1.07/2.3.00/20.0170 of the Ministry of Education, Youth and Sports of the Czech Republic).

References

1. Anselin, L.: GeoDa^{TM} 0.9 User's Guide. (2003).
2. Assuncao, R., Reis, E.: A new proposal to adjust Moran's I for population density. Stat. Med. 2162, November 1998, 2147–2162 (1999).
3. Beale, L. et al.: Methodologic issues and approaches to spatial epidemiology. Environ. Health Perspect. 116, 8, 1105–10 (2008).
4. Clayton, D., Bernardinelli, L.: Bayesian methods for mapping disease risk. In: Elliott, P. et al. (eds.) Geographical and Environmental Epidemiology: Methods for Small Area Studies. Oxford University Press, Oxford (1996).
5. Elliott, P., Wartenberg, D.: Spatial Epidemiology: Current Approaches and Future Challenges. Environ. Health Perspect. 112, 9, 998–1006 (2004).
6. Goodchild, M., Haining, R.: GIS and spatial data analysis: Converging perspectives. Pap. Reg. Sci. 44, 0, 1–26 (2004).
7. Griffith, D., Arbia, G.: Detecting negative spatial autocorrelation in georeferenced random variables. Int. J. Geogr. Inf. Sci. 24, 3, 417–437 (2010).
8. Kulldorff, M.: A spatial scan statistic. Commun. Stat. - Theory Methods. 26, 6, 1481–1496 (1997).
9. Kulldorff, M., Nagarwalla, N.: Spatial disease clusters: Detection and inference. Stat. Med. 14, 8, 799–810 (1995).
10. Last, J., Abramson, J.: A Dictionary of Epidemiology. Oxford University Press, USA (2001).
11. Lawson, A.B.: Bayesian Disease Mapping: Hierarchical Modeling in Spatial Epidemiology. CRC Press (2009).
12. Lee, D.: CARBayes: An R Package for Bayesian Spatial. J. Stat. Softw. 55, 13, 24 (2013).
13. Lee, D., Mitchell, R.: Boundary detection in disease mapping studies. Biostatistics. 13, 3, 415–26 (2012).
14. Marek, L. et al.: Bayesian mapping of medical data. (2014).
15. Marek, L. et al.: On Estimation of the Spatial Clustering: Case Study of Epidemiological Data In Olomouc Region, Czech Republic. VŠB – Technická univerzita Ostrava, Ostrava (2013).
16. Moran, P.: Notes on continuous stochastic phenomena. Biometrika. 37, 1, 17–23 (1950).
17. Openshaw, S. et al.: A Mark 1 Geographical Analysis Machine for the automated analysis of point data sets. Int. J. Geogr. Inf. Syst. 1, 4, 335–358 (1987).
18. Openshaw, S.: The Modifiable Areal Unit Problem. , Norwhich (1984).
19. Pfeiffer, D. et al.: Spatial analysis in epidemiology. Oxford University Press (2008).
20. Richardson, S. et al.: Interpreting Posterior Relative Risk Estimates in Disease-Mapping Studies. Environ. Health Perspect. 112, 9, 1016–1025 (2004).
21. Waller, L.: Detection of clustering in spatial data. SAGE Handb. Spat. Anal. 34 (2009).
22. Waller, L.A., Gotway, C.A.: Applied Spatial Statistics for Public Health Data. John Wiley & Sons (2004).
23. The Center for food security & public health: Campylobacteriosis, available at: http://www.cfsph.iastate.edu/FastFacts/pdfs/campylobacterosis_F.pdf, (2013).

Performance Analysis of the Activation Neuron Function in the Flexible Neural Tree Model

Tomáš Buriánek and Sebastián Basterrech

IT4Innovation
VŠB–Technical University of Ostrava,
Czech Republic
{Tomas.Burianek.St1,Sebastian.Basterrech.Tiscordio}@vsb.cz

Abstract. The time series prediction and forecasting is an important area in the field of Machine Learning. Around ten years ago, a kind of Multilayer Neural Network was introduced under the name of *Flexible Neural Tree (FNT)*. This model uses meta-heuristic techniques to determinate its topology and its embedded parameters. The FNT model has been successfully employed on time-series modeling and temporal learning tasks. The activation function used in the FNT belongs to the family of radial basis functions. It is a parametric function and the parameters are set employing an heuristic procedure. In this article, we analyze the impact on the performance of the FNT model when it used other family of neuron activation functions. For that, we use the *hyperbolic tangent* and *Fermi activation* functions on the tree nodes. Both functions have been extensively used in the field of Neural Networks. Moreover, we study the FNT technique with a linear variant of the Fermi function. We present an experimental comparison of our approaches on two widely used time-series benchmarks.

Keywords: Neural Network, Flexible Neural Tree, Neuron Activation Function, Time-series modeling, Forecasting

1 Introduction

The *Flexible Neural Tree (FNT)* have been used for several time-series problems as learning predictor. The model consists of an interconnected nodes forming a tree architecture [9, 10]. There are two kind of nodes, functional and terminal nodes. The terminal nodes contains the information of the input patterns. The functional nodes process the information using a specific activation function. The parameters of the model are: the weight connections among the nodes, the parameters in the activation function and the pattern of connectivity on the tree. The method combines two heuristic algorithms in order to find theses parameters. Several bio-inspired methods can be used as meta-heuristic technique to find the topology of the tree such as: Probabilistic Incremental Program Evolution (PIPE), Genetic Programing (GP), Ant Programming (AP). In order to find the embedded parameters, it can be used: Genetic Algorithms (GA), Particle Swarm Optimization (PSO) and Differential Evolution (DE), and so on.

The first FNT model was developed with a Gaussian activation function in the functional nodes.

In this paper we analyze the FNT performance when another kind of activation functions is used in the nodes. We study the nodes with hyperbolic tangent and Fermi activation function, both family of functions were extensively studied in the field of Neural Networks. Additionally, we test the model with a linear variant of the Fermi activation neurons. We present an experimental comparison of the different activation functions on two widely used time-series benchmarks. The first one is a simulated benchmark commonly used in the forecasting literature called 20-order fixed NARMA data set. The another one is a real data set about the Internet traffic from an European Internet service provider.

The paper is organized as follows. In Section 2, we present a description of the Flexible Neural Tree model. Section 3 contains the description of other activation functions. Next, we present our experimental results. Finally, the last part presents the article conclusion.

2 The Flexible Neural Tree model description

About ten years ago a new kind of *Neural Network (NN)* was introduced under the name of *Flexible Neural Tree (FNT)* [9, 10]. A FNT is a multilayer feedforward NN with an irregular topology. In the original FNT model each neuron has a parametric activation function. The network architecture is designed in an automatic way considering a pre-defined set of instructions and functional operators. The automatic process involves the parameters of the activation function, defines the pattern of connectivity among the units and selects the input variables. An evolutionary procedure is used to evaluate the performance of the tree topology. In temporal learning tasks is hard to select the proper input variables, the FNT technique uses an automatic procedure for this selection. Another meta-heuristic algorithm is employed to find the neural tree parameters. The FNT model proposes to use a simple random search method for setting the tree parameters. For this task can be use some of the following techniques:*Particle Swarm Optimization (PSO)* [13, 19] and *Differential Evolution (DE)* [16, 20]. In the pioneering FNT approach was used the Probabilistic Incremental Program Evolution (PIPE) [17] for encoding the NN in a tree [9]. In the literature other techniques were studied to find the topology and tree parameters, such that the *Genetic Programing (GP)* [4–6, 8] and *Ant Programming (AP)* [?].

2.1 The tree construction

The topology of the tree is generated using a pre-defined instruction set. We consider the following *function set* $\mathcal{F} = \{+_2, +_3, \ldots, +_{\mathrm{N_f}}\}$. A *node operator* $+_i$ is an internal vertex instruction with i inputs. In the original FNT, the activation neuron function of any unit i is the following parametric function:

$$f(a_i, b_i, x) = e^{-(\frac{x-a_i}{b_i})^2}, \tag{1}$$

where the parameters a_i and b_i are adjustable parameters. The *terminal set* of the tree consists of $\mathcal{T} = \{x_1, x_2, \ldots, x_{\mathrm{N_t}}\}$. The instruction set $\mathcal{S}$ is defined as follows:

$$\mathcal{S} = \mathcal{F} \cup \mathcal{T} = \{+_2, +_3, \ldots, +_{\mathrm{N_f}}\} \cup \{x_1, x_2, \ldots, x_{\mathrm{N_t}}\}. \tag{2}$$

Each functional node $+_i$ has i random nodes as its inputs, which can be internal and terminal nodes. The model outputs can be computed using a depth-first strategy for traversing the tree. Given a functional node $+_i$ with inputs $\{x_1, \ldots, x_i\}$, the total input charge of $+_i$ is defined as:

$$net_i = \sum_{j=1}^{i} w_j x_j, \tag{3}$$

where w_j is the weight connection between x_j and $+_i$. The output of the node $+_i$ is obtained using the expression (1)

$$\mathrm{out_i} = \mathrm{e}^{-(\frac{\mathrm{net_i - a_i}}{\mathrm{b_i}})^2}. \tag{4}$$

Let i be a functional node if some of its inputs are other functional nodes, then the output of i is computed in a recursive way following the expressions (3) and (4). The algorithm complexity for traversing the tree is $O(N)$ algorithmic time where N is the number of unit in the tree. Figure 1 illustrates an example of an FNT.

2.2 The fitness function

In order to evaluate the accuracy of the FNT model in learning tasks an error distance is considered. In this context, this performance measure is called *fitness function*. In a supervised learning problem given a training set $\{(\mathbf{x}(t), \mathbf{y}_{\mathrm{target}}(t)), t = 1, \ldots, T\}$, the goal is to infer a mapping $\varphi(\cdot)$ in order to predict $\mathbf{y}_{\mathrm{target}}$, such that the fitness function is minimized. Most often is used the *Mean Square Error* (MSE) or *Root Mean Square Error* (RMSE) [10], given by the expression:

$$MSE = \frac{1}{T}\sum_{t=1}^{T}(y(t) - \mathbf{y}_{\mathrm{target}}(t))^2. \tag{5}$$

Another important parameter of the model performance is the number of nodes in the tree. Obviously, between two trees with equal accuracy the smaller one is preferred.

2.3 The parameter optimization using meta-heuristic techniques

Several strategies have been used to find "good" parameters and topology [4, 7–10]. The model parameters embedded in the tree are the activation function parameters (a_i and b_i, for all $+_i$) and the weight connections. In this paper, we use Genetic Programming (GP) for finding a "good" connectivity among the

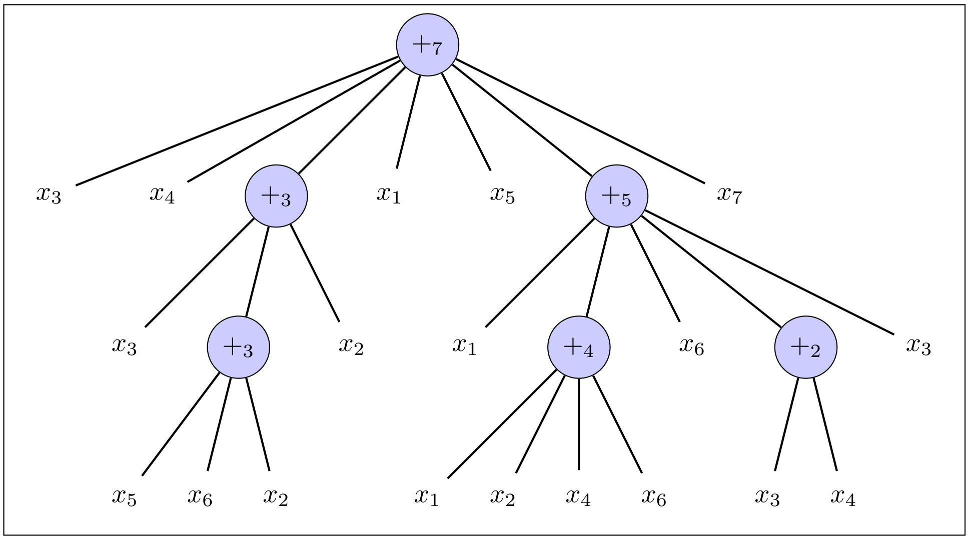

Fig. 1: Example of a Flexible Neural Tree with the function instruction set $\mathcal{F} = \{+_2, +_3, +_4, +_5, +_6, +_7\}$ and the terminal instruction set $\mathcal{T} = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$. The root in the tree $(+_7)$ is the output layer and the input layer is composed by the leaves in the bottom level of the tree.

neurons in the tree and we use the PSO method to find the embedded tree parameters.

The GP theory was intensely developed in the 90s [15]. A GP algorithm starts defining an initial population of same specific devices. The procedure is iterative, at each epoch it transforms a selected group of individuals producing a new generation. This transformation consists in applying some bio-inspired rules to the individuals. In our problem the individuals are the flexible trees. A set of individuals are probabilistically selected and a set of genetic rules is applied. The operating rules arise from some biological genetic operations, which basically consist of: *reproduction*, *crossover* and *mutation*. The reproduction is the identity operation, an individual i of a generation at time t is also presented at the generation at time $t+1$. Given two sub-tree the crossover consists in interchanging their parents. The mutation consists in selecting a tree and realizing one of the following operations: to change a leaf node to another leaf node, to replace a leaf node for a sub-tree, and to replace a functional node by a leaf node. The GP algorithms applied for our specific problem is described in Algorithm 1.

There are two kinds of adjustable parameters on the tree: the activation function parameters a_i and b_i for each functional node i presented in the expression (1), another one refers to the weight connections between the nodes in the tree. In this paper, we use *Particle Swarm Optimization (PSO)* [13] for finding these parameters. The PSO algorithm is an evolutionary computation technique

Algorithm 1: Specification of the Genetic Programming algorithm used for finding the optimal flexible tree topology.

```
   Inputs  : $N_t, N_f$, training data set, algorithm parameters
   Outputs: Tree
1  Initialize the population of trees;
2  Compute the fitness function for all trees;
3  while (Termination criterion is not satisfied) do
       // Creation of the new population
4      while (Size of population is not satisfied) do
5          Select genetic operation;
6          if (Selection is crossover) then
7              Select two trees from population using tournament selection;
8              Realize the crossover operation;
9              Insert the two new offspring into the new population;
10         if (Selection is mutation) then
11             Select one tree from population using tournament selection;
12             Realize the mutation operation;
13             Insert the new offspring into the new population;
14         if (Selection is reproduction) then
15             Select one tree from population;
16             Insert it to the new population;
17         Compute the fitness function of the new trees;
18     Replace old population by the new population;
19 Return the tree with best fitness function.
```

based on the social behaviors in a simplified social environment. A *swarm* is a set of particles, which are characterized by their position and their velocity in a multidimensional space. We denote the position of a particle i with the column N_x-vector $\mathbf{x}^{(i)} = (x_1^{(i)}, \ldots, x_{N_x}^{(i)})$. The velocity of i is defined by the column N_x-vector $\mathbf{ve}^{(i)} = (v_1^{(i)}, \ldots, v_{N_x}^{(i)})$. Besides, we use auxiliary vectors $\mathbf{p}^*$ and $\mathbf{p}^{(i)}$, $\forall i$, each one has dimension $N_x \times 1$. The vector $\mathbf{p}^{(i)}$ denotes the best position of i presented until the current iteration. The best swarm position is represented by the vector $\mathbf{p}^*$. Besides, we use two auxiliary random weights $\mathbf{r}_1^{(i)}$ and $\mathbf{r}_2^{(i)}$ of dimensions $N_x \times 1$, which are randomly initialized in $[0, 1]$ for each particle i. At any iteration t, the simulation of the dynamics among the particles is given by the following expressions [18]:

$$v_j^{(i)}(t+1) = c_0 v_j^{(i)}(t) + c_1 r_{1_j}(t)^{(i)}(p_j^{(i)}(t) - x_j^{(i)}(t)) + c_2 r_{2_j}(t)^{(i)}(p_j^*(t) - x_j^{(i)}(t)), j \in [1, N_x] \tag{6}$$

and

$$x_j^{(i)}(t+1) = x_j^{(i)}(t) + v_j^{(i)}(t+1), j \in [1, N_x], \tag{7}$$

where the constant c_0 is called the *inertia weight* the constants c_1 and c_2 regulate local and global position of the swarm, respectively.

In order to use PSO for estimating the embedded tree parameters, the position $\mathbf{p}^{(i)}$ of the particle i is associated with the embedded parameters in one flexible tree (the weights and a_j, b_j, $\forall j \in \mathcal{F}$). The PSO algorithm return the global optimal position according to the fitness function, presented in the expression (5). The relationship between the tree parameters and the particle position is given by:

$$(p_1^{(i)}, \ldots, p_{N_\mathrm{x}}^{(i)}) = (a_1, \ldots, a_{\mathrm{N_f}}, b_1, \ldots, b_{\mathrm{N_f}}, \mathbf{w}), \tag{8}$$

where $\mathbf{w}$ is a vector with the tree weights.

We present in the Algorithm 2 the PSO method used as meta-heuristic technique for finding the tree parameters.

Algorithm 2: Specification of the Particle Swarm Optimization used for finding the embedded tree parameters.

```
Inputs  : $N_\mathrm{x}$, number of particles, $\mathcal{S}$, training data set, algorithmic
          parameters
Outputs: Tree parameters
1  $t = 0$;
2  Random initialization of $\mathbf{p}^{(i)}(t)$ and $\mathbf{v}^{(i)}(t)$ for all $i$;
3  Compute the fitness value associated with $i$ using (8) and the fitness
   function;
4  Set $\mathbf{p}^*(t)$ and $\mathbf{p}^{(i)}(t)$ for all $i$;
5  while (Termination criterion is not satisfied) do
6      for (Each particle i) do
7          Compute $\mathbf{v}^{(i)}(t+1)$ using the expression (6);
8          Compute $\mathbf{x}^{(i)}(t+1)$ using the expression (7);
9          Compute the fitness value associated with $i$ using (8) and the
           fitness function;
10         Compute $\mathbf{p}^{(i)}(t+1)$;
11     Compute $\mathbf{p}^*(t+1)$;
12     t=t+1;
13 Return the parameters using $\mathbf{p}^*(t)$ and the expression (8);
```

3 Performance of other neuron activation functions in the nodes of the tree

In this paper we analyze the impact of other neuron activation function in the performance of the FNT model. The original model uses a Gaussian function presented in the expression (4). This kind of function has been widely used in Support Vector Machine (SVM) where the model uses the radial basis function (RBF) kernel [11]. Additionally it has been used in Self-Organizing Maps (SOM) as *neighbourhood function* among the neurons on the Kohonen networks [14]. In the area of Neural Network two kind of activation function have been extensively used: the tanh(·) function and the Fermi function. The sigmoid activation function is

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{9}$$

The Fermi activation function is

$$f(x) = \frac{1}{1 + e^{-x}}. \tag{10}$$

In this paper we test the FNT model using the tanh(·) and Fermi function. Moreover, we analyze a parametric variation of these functions, given by:

$$g(x) = a_i f(x) + b_i, \tag{11}$$

where $f(x)$ is the function of expression (9) and (10) and the parameters a_i and b_i are specifics for each tree node i. We use PSO in order to find the parameters a_i and b_i.

4 Empirical results

4.1 Description of the benchmarks

We use two benchmarks, a simulated data set which has been widely used in the forecasting literature and a real data set about the Internet traffic data.

(1) Fixed kth order NARMA data set. This data set presents a high non-linearity, for this reason has been extensively analized in the literature [1,3],

$$b(t+1) = 0.3b(t) + 0.05b(t)\sum_{i=0}^{k-1} b(t-i) + 1.5s(t-(k-1))s(t) + 0.1,$$

where $k = 20$ and $s(t) \sim Unif[0, 0.5]$. The task consists to predict the value $b(t+1)$ based on the history of $b(t)$ up to time t. The size of the training data was 3980 and the test data numbered 780.

(2) The Internet traffic data from United Kingdom Education and Research Networking Association (UKERNA). The data was collected from 19 November and 27 January, 2005. This data set was analyzed in [2, 12]. The problem was studied collecting the data in five minute scale, The goal is to predict the traffic at time t using the information of the Traffic in the time $\{t-1, t-2, t-3, t-4, t-5, t-6\}$. This sliding window was studied in [12]. The training data corresponds to the initial 66% of the data. The size of the training set is the 13126 and the test set has 6762 patterns. We normalized the data in $[0, 1]$.

4.2 Results

We can see in the two graphics of Figure 2 the performance of the FNT prediction for the NARMA test data set using Gaussian activation function and Fermi activation function. Figure 3 shows the prediction of the model for the NARMA data set using the Gaussian activation function. In this last case we present the estimation on the last 200 samples. A example of estimation of the FNT using Gaussian is showed in the left figure of 4. The right figure of 4 shows the estimation of the model when the activation function is Fermi function. A example of prediction of the FNT model using the Fermi activation function for the Internet traffic data set is illustrated in Figure 5. We can see in Figure 4 the FNT estimation with the Fermi activation and the Gaussian activation function are very close. The Fermi function is not parametric, then the PSO algorithm is used only to estimate the weights in the tree. As a consequence the FNT model with Fermi function is faster in the training process than the FNT with exponential parametric function. The accuracy for the Internet Traffic data using the FNT technique and Fermi activation function was better than when we used the Gaussian activation function. Table 1 shows a comparison of the accuracy of the model using the different kinds of activation function in the nodes.

Function	Narma	Internet Traffic data
Gaussian	2.25646×10^{-3}	10.0642×10^{-5}
$\tanh(\cdot)$	3.47218×10^{-3}	10.2117×10^{-5}
Fermi function	2.37082×10^{-3}	8.67417×10^{-5}
Linear Fermi	3.23204×10^{-3}	9.95649×10^{-5}

Table 1: Accuracy of the FNT models for different kind of activation function in the tree nodes. The first and second column show the MSE obtained by the model, the error is presented using a scientific notation. First row refers the function used in the original FNT. The last three rows refer to the functions 9, 10 and 11.

5 Conclusions and future work

Ten years ago, the *Flexible Neural Tree (FNT)*, a specific type of Neural Network was presented. The method has proved to be a very powerful tool for time series processing and forecasting problems. The model uses meta-heuristic techniques for defining the topology of the tree and for finding “good” parameters in learning tasks. The FNT uses activation function with exponential form in the nodes. In this paper we analyze the performance of the model with another family of function in its nodes, we studied the hyperbolic tangent and the Fermi functions. We tested the performance in two widely used benchmark data: a simulated and a real data set. The results are promising, specifically for the FNT model that uses Fermi function its the nodes. In future works, we will use statistical tests for comparing the different approaches presented here, as well as we will compare our results with other forecasting techniques.

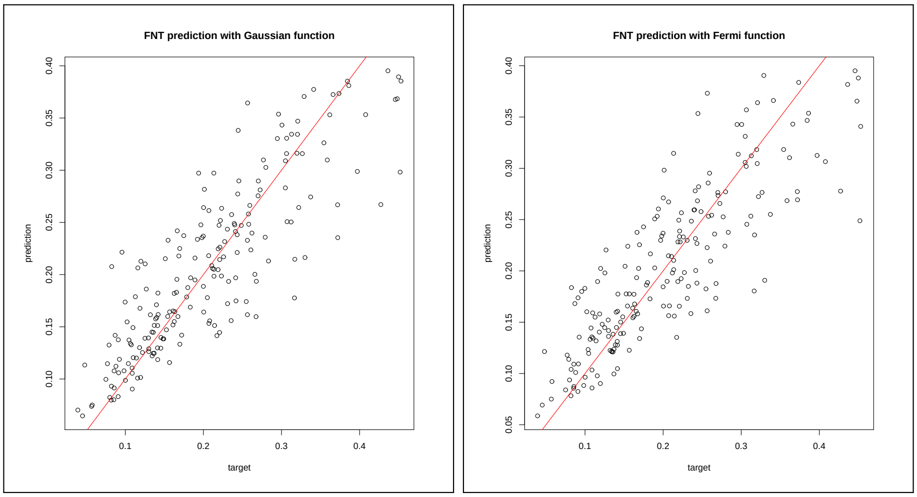

Fig. 2: Example of the FNT prediction for the NARMA data set. Both figures show the estimation of the last 80 time units of the test data set. The left figure shows the FNT prediction using the Gaussian activation function. The right figure illustrates the estimation of the model using the Fermi activation function. The red line corresponds the identity function.

Acknowledgments

This article has been elaborated in the framework of the project *New creative teams in priorities of scientific research*, reg. no. CZ.1.07/2.3.00/30.0055, supported by Operational Program Education for Competitiveness and co-financed by the European Social Fund and the state budget of the Czech Republic. Additionally, this work was partially supported by the Grant of SGS No. SP2014/110, VŠB - Technical University of Ostrava, Czech Republic, and was supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070) and by the Bio-Inspired Methods: research, development and knowledge transfer project, reg. no. CZ.1.07/2.3.00/20.0073 funded by Operational Programme Education for Competitiveness, co-financed by ESF and state budget of the Czech Republic.

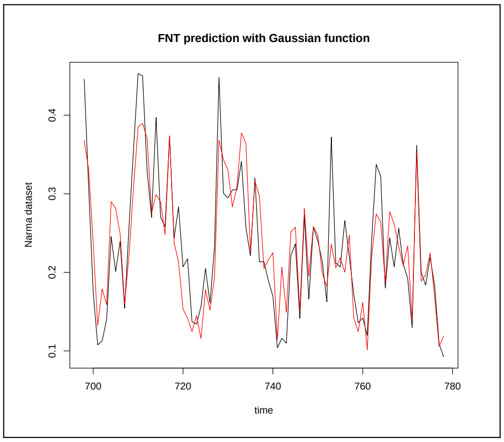

Fig. 3: FNT prediction using the Gaussian activation function on the NARMA data set. The estimation was realized on the last 200 time units of the test data. The red line is the prediction o the data and the black line is the target data.

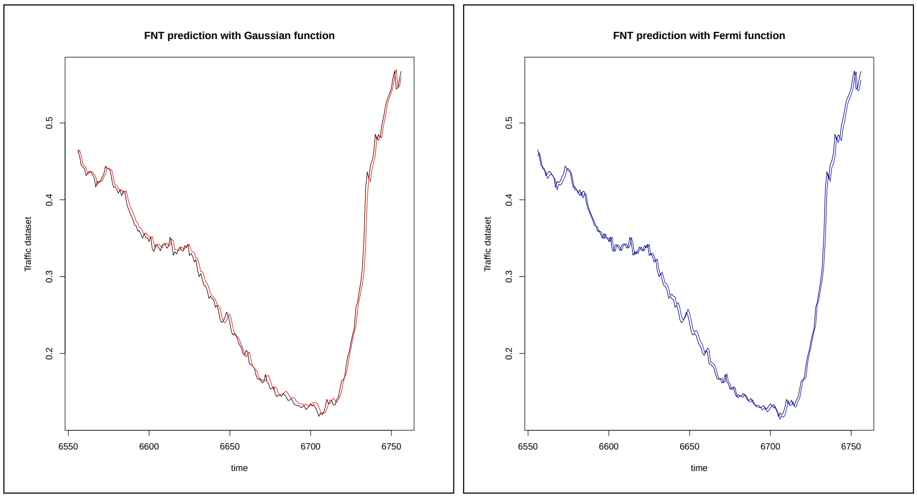

Fig. 4: FNT prediction on the Internet traffic dataset. Both figures show the estimation of the last 200 time units of the test data set. The left figure shows the prediction with the Gaussian activation function (red line). The right figure illustrates the estimation of the model (blue line) using the Fermi activation function. In both cases the black line is the target data.

References

1. Sebastián Basterrech, Colin Fyfe, and Gerardo Rubino. Self-Organizing Maps and Scale-Invariant Maps in Echo State Networks. In *Intelligent Systems Design and*

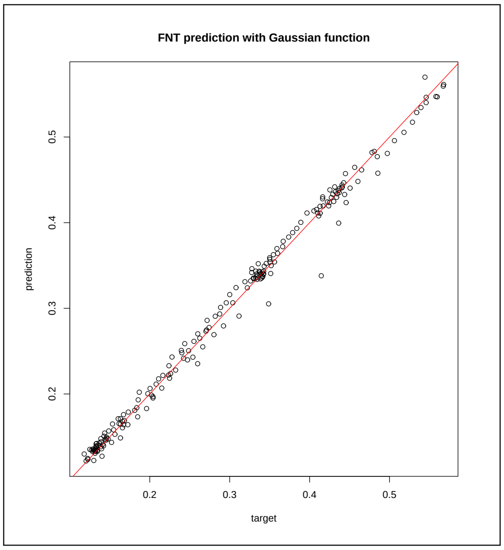

Fig. 5: FNT prediction using the Gaussian activation function on the Internet traffic data set. The estimation was realized on the last 80 time units of the test data set. The red line is the identity function.

Applications (ISDA), 2011 11th International Conference on, pages 94–99, nov. 2011.

2. Sebastián Basterrech and Gerardo Rubino. Echo State Queueing Network: a new Reservoir Computing learning tool. *IEEE Consumer Comunications & Networking Conference (CCNC'13)*, pages 118–123, January 2013.
3. Sebastián Basterrech and Václav Snášel. Initializing Reservoirs With Exhibitory And Inhibitory Signals Using Unsupervised Learning Techniques. In *International Symposium on Information and Communication Technology (SoICT)*, pages 53–60, Danang, Viet Nam, December 2013. ACM Digital Library.
4. Yuehui Chen, Ajith Abraham, and Bo Yang. Feature selection and classification using flexible neural tree. *Neurocomputing*, 70(1-3):305–313, 2006.
5. Yuehui Chen, Ajith Abraham, and Bo Yang. Hybrid flexible neural-tree-based intrusion detection systems. *International Journal of Intelligent Systems*, 22(4):337–352, 2007.
6. Yuehui Chen, Ajith Abraham, and Ju Yang. Feature selection and intrusion detection using hybrid flexible neural tree. In Jun Wang, Xiao-Feng Liao, and Zhang Yi, editors, *Advances in Neural Networks*, volume 3498 of *Lecture Notes in Computer Science*, pages 439–444. Springer Berlin Heidelberg, 2005.
7. Yuehui Chen, Lizhi Peng, and Ajith Abraham. Exchange rate forecasting using flexible neural trees. In Jun Wang, Zhang Yi, JacekM. Zurada, Bao-Liang Lu, and Hujun Yin, editors, *Advances in Neural Networks - ISNN 2006*, volume 3973 of *Lecture Notes in Computer Science*, pages 518–523. Springer Berlin Heidelberg, 2006.

8. Yuehui Chen, Bo Yang, and Ajith Abraham. Flexible neural trees ensemble for stock index modeling. *Neurocomputing*, 70(4-6):697–703, 2007.
9. Yuehui Chen, Bo Yang, and Jiwen Dong. Nonlinear System Modelling Via Optimal Design of Neural Trees. *International Journal of Neural Systems*, 14(02):125–137, 2004.
10. Yuehui Chen, Bo Yang, Jiwen Dong, and Ajith Abraham. Time-series forecasting using flexible neural tree model. *Inf. Sci.*, 174(3-4):219–235, August 2005.
11. Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Mach. Learn.*, 20(3):273–297, September 1995.
12. P. Cortez, M. Rio, M. Rocha, and P. Sousa. Multiscale Internet traffic forecasting using Neural Networks and time series methods. *Expert Systems*, 2012.
13. J. Kennedy and R. Eberhart. Particle Swarm Optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948, 1995.
14. Teuvo Kohonen. *Self-Organizing Maps.* Springer Series in Information Sciences, third edition, 2001.
15. John R. Koza, Forrest H. Bennett III, and Oscar Stiffelman. Genetic Programming as a Darwinian Invention Machine. In Riccardo Poli, Peter Nordin, WilliamB. Langdon, and TerenceC. Fogarty, editors, *Genetic Programming*, volume 1598 of *Lecture Notes in Computer Science*, pages 93–108. Springer Berlin Heidelberg, 1999.
16. Kenneth Price, Rainer M. Storn, and Jouni A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.
17. Rafal Salustowicz and Jürgen Schmidhuber. Probabilistic incremental program evolution. *Evolutionary Computation*, 5(2):123–141, 1997.
18. Yuhui Shi and R. Eberhart. A modified particle swarm optimizer. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pages 69–73, 1998.
19. Yuhui Shi and RussellC. Eberhart. Parameter Selection in Particle Swarm Optimization. In V.W. Porto, N. Saravanan, D. Waagen, and A.E. Eiben, editors, *Evolutionary Programming VII*, volume 1447 of *Lecture Notes in Computer Science*, pages 591–600. Springer Berlin Heidelberg, 1998.
20. Rainer Storn and Kenneth Price. Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

Ternary Tree Optimalization for n-gram Indexing

Daniel Robenek, Jan Platoš, Václav Snášel

Department of Computer Science, FEI, VSB – Technical University of Ostrava,
17. listopadu 15, 708 33, Ostrava-Poruba, Czech Republic
{daniel.robenek, jan.platos, vaclav.snasel}@vsb.cz

Abstract. N-gram indexing is used in many practical applications. Spam detection, plagiarism detection or comparison of DNA reads. There are many data structures that can be used for this purpose, each with different characteristics. In this article the ternary search tree data structure is used. One improvement of ternary tree that can save up to 43% of required memory is introduced. In the second part new data structure, named ternary forest, is proposed. Efficiency of ternary forest is tested and compared to ternary search tree and two-level indexing ternary search tree.

Keywords: n-gram, ternary tree, ternary forest, inverted index

1 Introduction

Efficient indexing and searching in huge amount of data is big issue in computer science. For example finding plagiarisms, spam detection or comparison of DNA sequences are topics, where efficient indexing is a key element of fast software.

The piece of data in these problems can be called n-gram. For DNA sequences n-grams are nucleotides in the sequence read. For plagiarism and spam detection n-grams are words in sentences or characters in the words.

First problem is to efficiently extract these n-grams. There are many specific and optimized algorithms for this purpose. Next is necessary to perform the indexing.

Many data structures for this purpose, with different efficiency in search, insertion or memory requirements are known. Indexing can be divided into two main categories, depending on available memory or amount of data. First, when amount of the data exceeds available memory, the data are stored on hard drive. Data structures like B+ tree are optimized for this purpose.

Second category is in-memory based indexing, which expects sufficient amount of memory for this purpose. The article is mainly focused to second category, specifically to ternary tree optimization for n-gram indexing.

2 Related Work

The path from plain text documents to ready-made search engine is long and difficult. The first problem is to extract required n-grams out of documents into required format. In case of spam detection or plagiarism detection we are interested in n-grams that occur at last m-times [8]. It is because we are comparing similarity of documents, respectively the most repeated parts of them.

In case of extremely large amount of documents the common data structures like hash tables or trees are not suitable, because they would probably not fit in the memory. Therefore another approach is inevitable. By using sophisticated algorithms and hard drive as a temporary storage high efficiency can be achieved without necessity of having large amount of RAM [4]. For smaller sets of data the utilized data structures can be used [6].

A number of data structures have been proposed for text search and inverted index is the most used one [1]. For example inverted index is used to evaluate queries in many search engines [7]. The optimization and compression can be used on inverted index data structures in order to speed up the search and reduce memory requirements [10,5].

There are many data structures that can do n-gram indexing in memory [9]. One opportunity how to create in memory inverted index is to use ternary search tree in which every node stores information about one n-gram character. As it was shown by tests on collections Google WebIT and English Gigaword corpus, the data structure is fast enough [3].

However storing the whole n-gram into single data structure as-is may not be optimal. Repeated words should not be stored many times. Redundant presence of words causes excessive memory consumption. The idea is to create two data structures where the words in n-grams are at first converted to unique numbers and only after that the numbers are processed by data structure [6,7].

Using two-level inverted index can considerably decrease memory consumption [9,2].

One of the requirements for these types of data structures is the opportunity to use wildcard placeholders. This is used when is necessary to look for particular similarity. When using two-level inverted index it is necessary to find range of words on the first index. However this is only efficient when indexes are sorted with the words. When this request is fulfilled, it is easy to look up for these words using data structures like B+ tree [2].

3 Tree Compression

When using general binary or ternary tree, every unigram is contained in separate part of tree named node. The node contains necessary tree parts, parts that are specific for different kinds of each type of the tree and the unigram key and n-gram value.

Unigram key can be represented by *char* data type in case of indexing words. The pointer can be also used when needed. Situation is similar for n-gram value. When we are not interested in value, for example when use tree as a data set structure, the value can be omitted.

Data are partly specific to different tree implementations. Red-black tree needs to contain color value and each node of AVL tree should contain depth information. These information are necessary for self-balancing efficiency.

Last parts of tree node are references to next tree nodes. For binary tree, two pointers are needed. For ternary tree, one more pointer is needed. The question is, if these references are necessary. Binary tree nodes can be divided into two types - internal nodes and external nodes (leaf nodes). Internal nodes are those having at last one child. On the other hand, external nodes have no children.

In following sections, the ternary red-black tree will be used. New method for unused ternary tree references removal will be introduced and results of test will be shown.

3.1. Binary Tree

For memory saving computation is necessary to exactly define how both internal and external node should look like. Comparison is shown in Table 1. For computations the key variable is 8-bit character type and the value is 32-bit integer. References are also 32-bit integers.

Table 1. Composition of binary tree nodes

Variable	Internal node	External node
Left reference	4 bytes	0 bytes
Right reference	4 bytes	0 bytes
Color of node	1 byte	0 bytes
Key	1 byte	1 byte
Value	4 bytes	4 bytes
Sum of size	14 bytes	5 bytes
Real size	16 bytes	8 bytes

The size of external node in extreme case is only ~36% of internal node. The real test showed different values. Because of memory alignment, size of internal node is 16 bytes and external node size is 8 bytes. But the 50% is still large difference. The real size may vary, depending on the *Key* and *Value* variable sizes.

To compute memory saves of tree is necessary to know amount of internal and external nodes. Lets think about ideal binary tree, where every node has zero or two child nodes. The relative number of internal is:

$$c_{ne} = 2^{h-1}$$

$$c_{ni} = 2^{h-1} - 1$$

$$c_{ne} = c_{ni} + 1$$

Where c_{ne} is amount of external nodes, c_{ni} is amount of internal nodes and h is height of tree. If we consider only large trees, we can assume that

$$c_{ne} = c_{ni}$$

with negligible error. Now, for this type of tree and sizes of nodes mentioned in Table 1, we can compute memory saving for whole tree. Without optimization, the size of the node is 14 bytes. With optimization, the size of average node is ~9.5 bytes. It means ~32% of saved memory.

If we consider also memory alignment, the results are little different. Node size is 16 bytes and with optimization, average node size is 12 bytes. It results in 25% less memory usage.

This was one, ideal type of binary tree. But we can generalize these results for every binary tree. The amount of used references in binary tree is always same, no matter of tree arrangement. This amount is exactly same as number of nodes minus one, because root node has no reference to its parent node.

If we remove every unnecessary reference in the tree, the real saved memory would be about 25%, depending on *Key* and *Value* data type.

3.2. Ternary Tree

On Table 2 we can see node sizes of ternary search tree.

Table 2. Composition of ternary tree nodes

Variable	**Internal node**	**External node**
Left reference	4 bytes	0 bytes
Right reference	4 bytes	0 bytes
Middle reference	0 / 4 bytes	0 / 4 bytes
Color of node	1 byte	0 bytes
Key	1 byte	1 byte
Value	4 bytes	4 bytes
Sum of size	14 / 18 bytes	5 / 9 bytes
Real size	16 / 20 bytes	8 / 12 bytes

The difference is middle reference, which can enlarge node size of 4 bytes. Therefore minimum size of node is 5 bytes for leaf with no middle reference, and maximum is 16 bytes for internal node with middle reference or 8 bytes and 20 bytes for real allocated size.

If root node reference is omitted, there are 4 bytes per node for reference and 6 bytes for data. Therefore average amount of memory for one node of ternary tree is 10 bytes. This is ~44% less memory usage compared to tree created of nodes with complete references.

The memory saving can be increased by removing unused value variable.

3.3. Ternary Tree Tests

To prove these computations and to get time requirements of this optimization the tests were performed. The tests were performed on two data structures, which are previously mentioned compressed red-black ternary tree and ordinary red-black ternary tree. Every data structure was tested with predefined set of n-grams, from 1,000,000 to 100,000,000 each. These n-grams have average length of 11 characters. Moreover the efficiency of these structures was tested on different size of n-grams. The tests were performed on computer with 84xE5-4610@2,4GHz processor with 1 Tb of RAM. N-grams were extracted from Web 1T 5-gram, 10 European Languages Version 1 collection.

To simplify implementation and not to slow down search and insertion too much the implementation for tests counts only with two types of nodes. They were performed with common internal node and with external node without left and right reference. Each of these nodes has its own type definition in code and the resolution of the node type is done by the node index. As an alternative, one bit identifier may be also used. Transformation from red-black tree into compressed red-black tree is made by post processing.

3.3.1. Search Time

On the Figure 1, there is comparison between search times of compressed red-black tree and common red-black tree. Graph shows slightly decreasing trend with average slowdown of 3% amount. This slowdown is caused by type check of node on access. This amount of slowdown seems to be acceptable in comparison of theoretical memory saving.

Note that amount of n-grams on Figure 1 do not increase linearly.

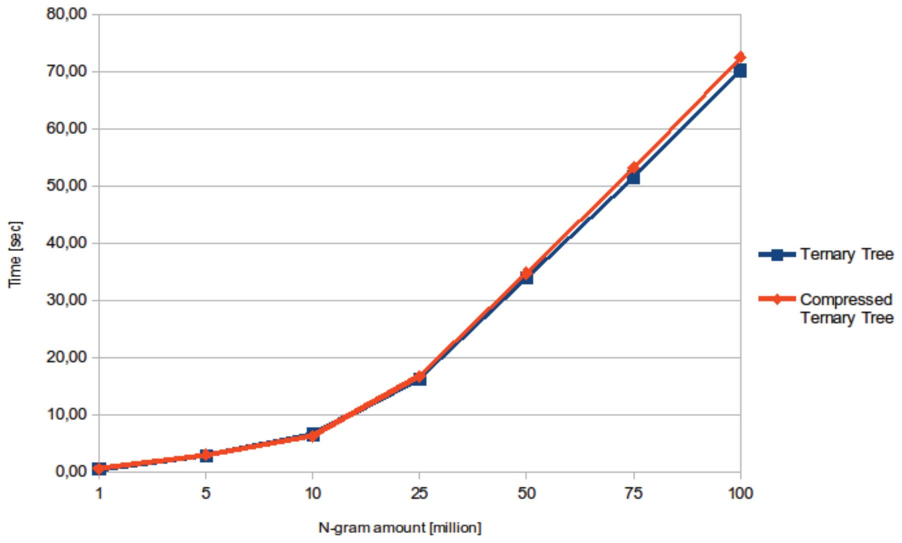

Figure 1. Search time comparison

3.3.2. Insertion Time

Comparison of insertion time is shown on Figure 2. This graph shows also slightly decreasing trend. The average slowdown is 16%. This number may be too high, but for many applications is more relevant search time or memory usage.

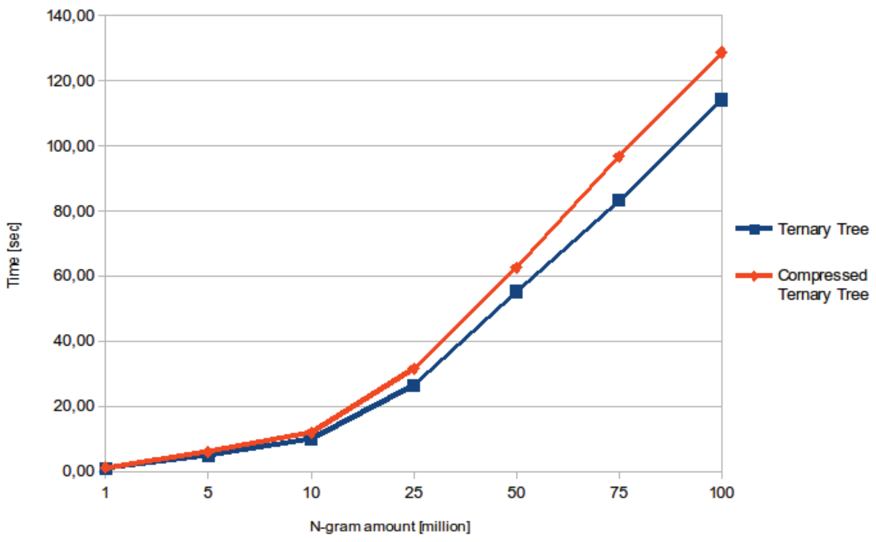

Figure 2. Insertion time comparison

3.3.3. Memory Usage

The Figure 3 shows memory usage of compared tree structures. Memory savings seems to be stable, about 35%. This amount of saved memory is high when we consider that the implementation does not remove middle reference, single left, or single right reference.

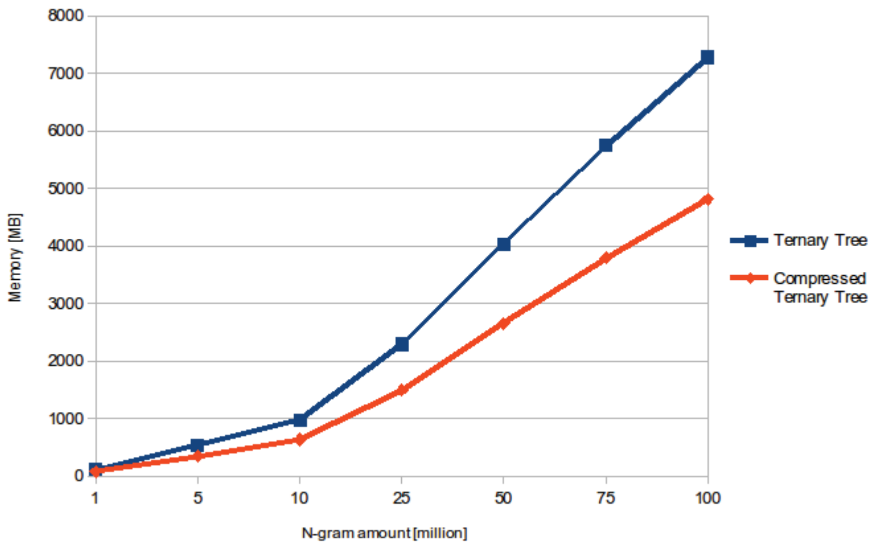

Figure 3. Memory consumption comparison

To explain this behavior is necessary to count amount of internal and external nodes of tested tree. Results show that almost 90% of all nodes are external nodes, nodes with no left or right child. The amount of nodes without middle reference is about 17%. This seems to be negligible for removal.

The amount of nodes with value reference is about 82%. Therefore removal of this type of reference can improve memory saving even more, especially when variable size is larger than 4 bytes. By modifying external node implementation to have no left reference, right reference and value variable the memory saving raises to approx. 43% without substantial slowdown.

3.4. Tests with n-gram Size

In previous chapters the behavior of compressed red-black tree depending on amount of n-grams was tested. The question is how the size of n-grams affects compressed n-gram tree performance.

Figure 4 shows search time with different size of n-grams from ~12 to ~24 characters. The amount of n-grams is 25,000,000. Results shows only small differences compared with common red-black ternary tree, with better results on greater n-gram size.

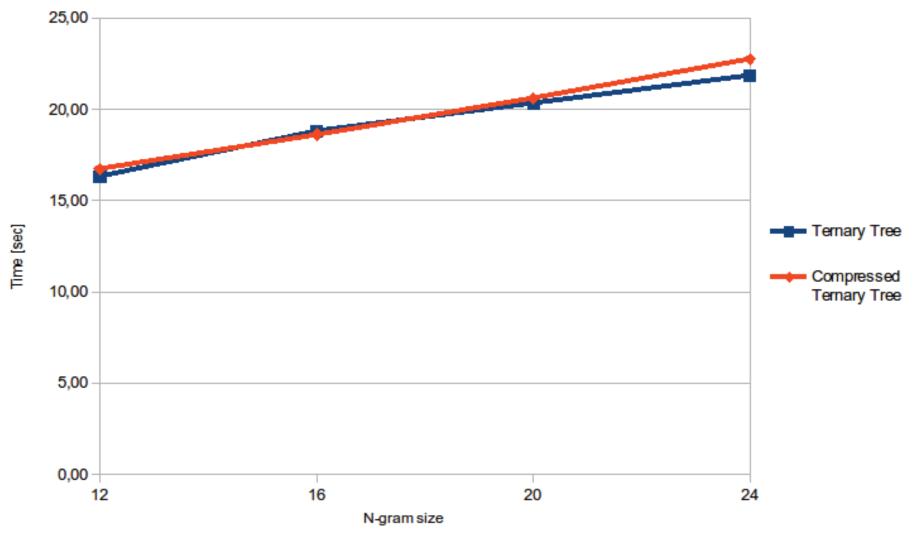

Figure 4. Search time comparison

Differences in insertion time show Figure 5. The time necessary to create and fill compressed ternary tree rises with n-gram size. This behavior may be caused by recursive algorithm used in tree compression.

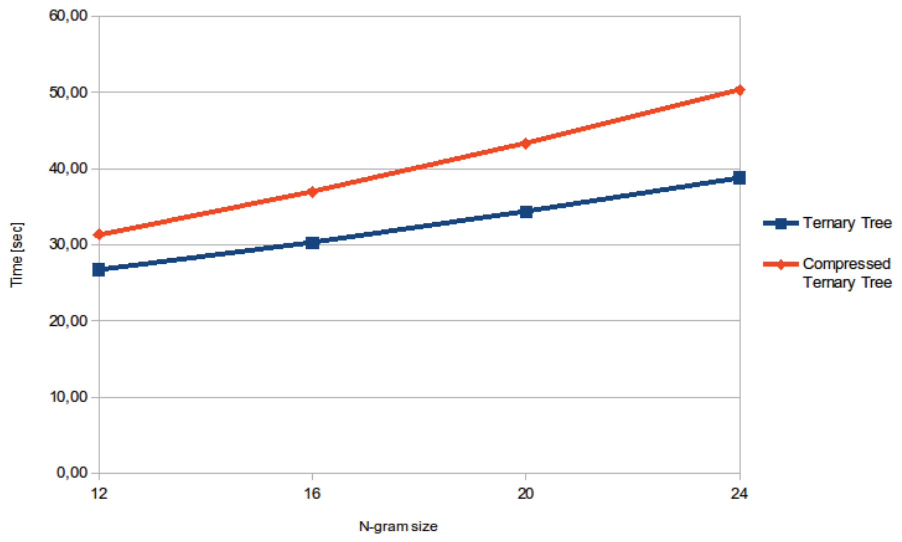

Figure 5. Insertion time comparison

Figure 6 shows memory requirements. Compressed red-black ternary tree has greater memory saving on longer n-grams. Increasing amount of single node binary trees in ternary tree causes this.

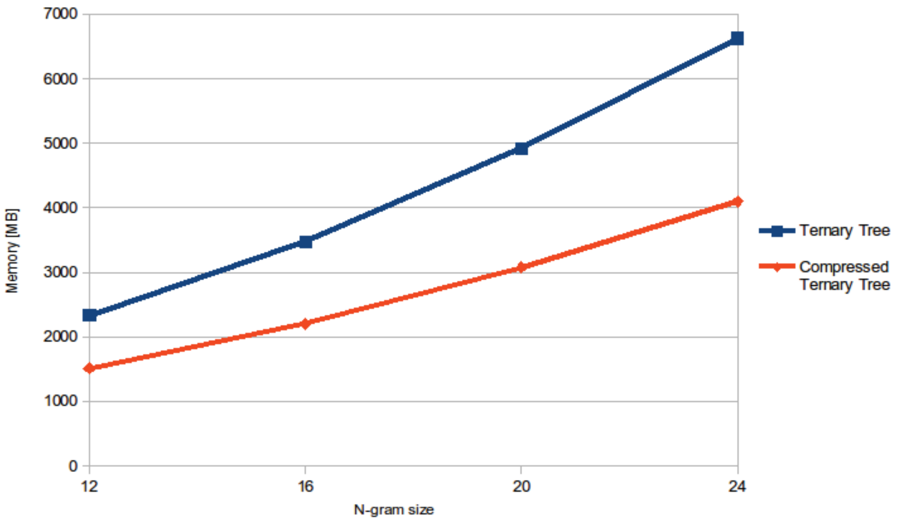

Figure 6. Memory consumption comparison

4 Ternary Forest

For sentence indexing is appropriate to use two-level (double) indexing. This approach saves a lot of computer memory, because words in all sentences are many times repeated.

Common ternary tree n-gram indexing and n-gram double indexing using ternary trees are two borderline cases. Single ternary tree is much faster in search and insertion time. This is mainly caused by low height of binary trees deeper in the ternary tree. Unfortunately this approach requires lot of computer memory.

Double indexing can save large amount of required memory, because it reduces duplicities in the tree. Disadvantage of this approach is slowdown in both search and insertion time, because deeper binary trees takes more time to be searched. But can we combine both approaches to get more balanced data structure?

One approach can be ternary forest. Ternary forest is created from two types of ternary search trees. First, *word* tree is indexing characters of words and second, *n-gram tree* is indexing whole words. Every of the second trees are connected to the last node of the first tree.

On Figure 7 is shown how three words "AB AC AB" can be indexed. In the *word tree*, node *A* is first binary tree. Second binary tree consists of nodes *B* and *C*. Second part of structure *n-gram tree* consists of two more single node binary trees, these are with keys *3* and *2*.

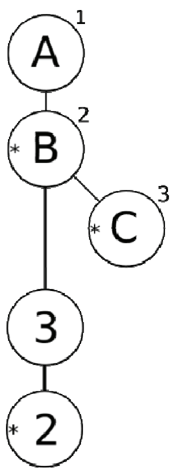

Figure 7. Ternary forest example

To search words "AB AC AB" is necessary to find first word in first part of data structure named *word tree*. When the first word is found, reference to the second part of the data structure *n-gram tree* is stored. Then the second word "AC" is found in the *word tree* with result *3*. The stored root index of *n-gram tree* is used to found node with index 3. Search is done again in the *word tree* with index *2* and the last node in the *n-gram tree* is found.

Advantage of this approach is that indexing trees are not separated, but second n-gram tree is directly connected to first *word* tree. This little difference from common double indexing may look negligibly.

The test was performed to show depth of binary trees in n-gram tree. The set of 10,000,000 5-grams was used. The results shown that over 90% of trees are single node trees. But more important is, that root tree has depth of size 32. Using ternary forest instead of double indexing can rapidly reduce this size.

Moreover, sequence amount of words in the word tree can differ. On Figure 7 word tree covers single word. But this may not be optimal for all purposes.

4.1. Ternary Forest Tests

Figure 8 shows behavior of insertion time, search time and memory requirements of data structures. First data structure is double indexing ternary search tree. Second is ternary forest, and the last one is common red-black ternary tree.
Data used for tests was 5-grams with average length of 24 characters. Ternary forest is used in four tests. In each test the ternary forest has different amount of words sequentially stored in the word tree in a row.

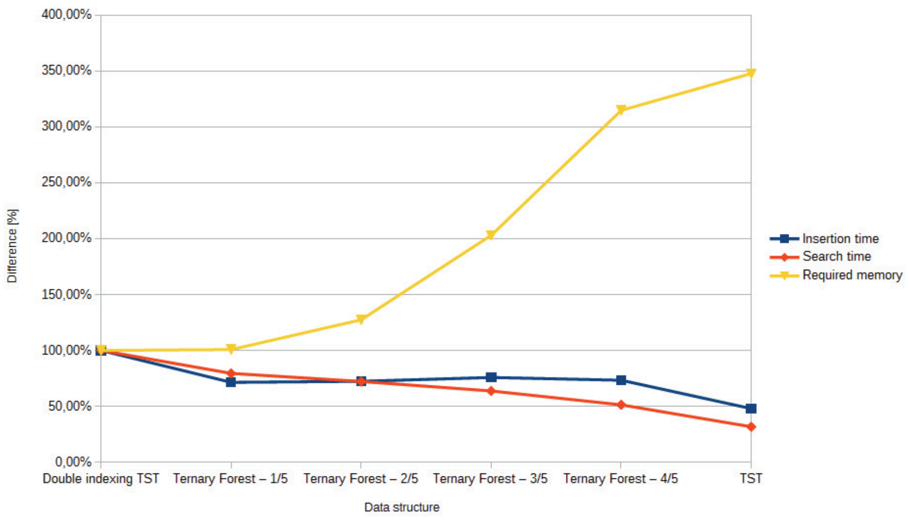

Figure 8. Relative comparison of double indexing ternary search tree (left), ternary forest and ternary search tree (right)

The results have shown that ternary forest using 1 word indexing in word tree has greatly improved performance. Insertion time speedup is ~30% and search time is ~20% faster than double indexing. The memory requirements show only negligible increase, less than 1%.

5 Conclusion

This paper described two improvements on ternary tree for efficient n-gram indexing. First, ternary tree compression showed how to save up to 43% of computer memory by removing unused references, without major slowdown.

Second improvement was more focused on n-gram indexing as such. By using ternary forest instead of common two-level indexing search time has decreased ~20% and insertion time ~30% with negligible increase of memory requirements.

Acknowledgement: This work was partially supported by the Grant of SGS No. SP2014/110, VŠB - Technical University of Ostrava, Czech Republic, and was supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070) and by the Bio-Inspired Methods: research, development and knowledge transfer project, reg. no. CZ.1.07/2.3.00/20.0073 funded by Operational Programme Education for Competitiveness, co-financed by ESF and state budget of the Czech Republic.

6 References

1. Baeza-Yates, Ricardo, and Berthier Ribeiro-Neto. *Modern information retrieval*. Vol. 463. New York: ACM press, 1999.
2. Ceylan, Hakan, and Rada Mihalcea. "An Efficient Indexer for Large N-Gram Corpora." *ACL (System Demonstrations)*. 2011.
3. Flor, Michael. "Systems and Methods for Optimizing Very Large N-Gram Collections for Speed and Memory." U.S. Patent Application 13/168,338, 2011.
4. Huston, Samuel, Alistair Moffat, and W. Bruce Croft. "Efficient indexing of repeated n-grams." *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM, 2011.
5. Kim, Min-Soo, et al. "n-gram/2l: A space and time efficient two-level n-gram inverted index structure." *Proceedings of the 31st international conference on Very large data bases*. VLDB Endowment, 2005.
6. Kratky, M., et al. "Index-based n-gram extraction from large document collections." *Digital Information Management (ICDIM), 2011 Sixth International Conference on*. IEEE, 2011.
7. MOFFAT, ALISTAIR AUTOR, and Timothy C. Bell. *Managing gigabytes: compressing and indexing documents and images*. Morgan Kaufmann, 1999.
8. Pomikálek, Jan, and Pavel Rychlý. "Detecting Co-Derivative Documents in Large Text Collections." *LREC*. 2008.
9. Robenek, Daniel, Jan Platoš, and Václav Snášel. "Efficient in-memory data structures for n-grams indexing."
10. Scholer, Falk, et al. "Compression of inverted indexes for fast query evaluation."*Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2002.

EEG signals similarity based on compression

Michal Prílepok, Jan Platoš, and Václav Snášel

Department of Computer Science, FEECS
IT4 Innovations, European Center of Excellence
VSB-Technical University of Ostrava
Ostrava, Czech Republic
{michal.prilepok, jan.platos, vaclav.snasel}@vsb.cz

Abstract. The electrical activity of brain or EEG signal is very complex data system that may be used to many different applications such as device control using mind. It is not easy to understand and detect useful signals in continuous EEG data stream. In this paper, we are describing an application of data compression which is able to recognize important patterns in this data. The proposed algorithm uses Lempel-Ziv complexity for complexity measurement and it is able to successfully detect patterns in EEG signal.

Keywords: Electroencephalography; EEG; BCI; EEG waves group; EEG data; LZ Complexity

1 Introduction

The Electroencephalography (EEG) plays a big role in diagnosis of brain diseases, and, also, in Brain Computer Interface (BCI) system applications that helps disabled people to use their mind to control external devices. Both research areas are growing today.

The EEG records the electrical activity of the brain using several sensors placed on a scalp . Different mental tasks produce indiscernible recordings but they are different. Different brain actions activate different parts of the brain. The most difficult part is the definition of an efficient method or algorithm for detection of the differences in recordings belonging to the different mental tasks. When we define such algorithm we are able to translate these signals into control commands of an external device, e.g. prosthesis, wheelchair, computer terminal, etc.

2 The Electroencephalography

The Electroencephalography (EEG) measures the electrical activity of human brain, by placing set of sensors on a scalp, according to 10/20 EEG International electrode placement, as is depicted on Figure 1. The measuring of EEG signal records can be done between two active electrodes (bipolar recording), or between an active electrode and a reference electrode (mono-polar recording) [16].

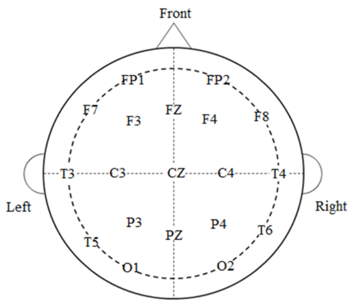

Fig. 1. 10/20 International Electrode Placement

2.1 EEG Waves Types

The types of brain waves distinguished by their different frequency ranges are recognized as follows.

– Delta (δ) waves lie within the range from approximately 0.5 up to 4 Hz. The amplitude of this waves is varying and have been associated with deep sleep and present in the waking state.
– Theta (θ) waves lie within the range from 4 to 7.5 Hz. The amplitude varies about 20 μV. Theta waves have been associated with access to unconscious material, creative inspiration and deep meditation.
– The frequency of the Alpha (α) waves lies within the range from 8 to 13 Hz, the amplitude varies between 30 and 50 μV. It is reduced or eliminated by opening the eyes, by hearing unfamiliar sounds, by anxiety, or mental concentration or attention.
– Beta (β) waves are the electrical activities of the brain varying within the frequency range from 14 to 26 Hz. The amplitude is about 5 up to 30 μV. Beta waves has been associated with active thinking, active attention, focus on the outside world, or solving concrete problems. A high-level beta wave may be acquired when a human is in a panic state.
– Gamma (γ) waves have frequency range above 30 Hz, can be used to demonstrate the locus for right and left index finger movement, right toes, and the rather broad and bilateral area for tongue movement [19, 18].
– Mu (μ) waves will be same Alpha frequency range 8 to 13 Hz, but Alpha waves are recorded on occipital cortex area, and Mu waves are recorded on motor cortex area. Mu waves are related to spontaneous nature of the brain such motor activities [18].

2.2 History of EEG

Carlo Matteucci and Emil Du Bois-Reymond, were first people who register the electrical signals emitted from muscle nerves using a galvanometer and established the concept of neurophysiology. The first brain activity in the form of electrical signals was recorded in 1875, by Richard Caton (1842–1926), a scientist from Liverpool, England, using a galvanometer and two electrodes placed over the scalp of a human. From here EEG stand to, Electro that referring to registration of brain electrical activities, Encephalon that referring to emitting the signals from a brain, and gram or graphy, which means drawing. Then the term EEG was henceforth used to denote electrical neural activity of the brain [19].

In 1920, Hans Berger, the discoverer of the existence of human EEG signals, began his study of human EEG. In 1910, Berger started working with a string galvanometer and later he used a smaller Edelmann model. After the year 1924, he used larger Edelmann model. Berger started to use the more powerful Siemens double coil galvanometer (attaining a sensitivity of 130 μ V/cm) in 1926. In 1929 Berger made the first report of human EEG recordings with duration from one to three minutes on photographic paper and, in the same year, he also found some correlation between mental activities and the changes in the EEG signals [19].

The first biological amplifier for the recording of brain potentials was built by Toennies (1902–1970). In 1932 the differential amplifier for EEG recording was later produced by the Rockefeller foundation. The potential of a multichannel recordings and a large number of electrodes to cover a wider brain region was recognized by Kornmuller. Berger assisted by Dietch (1932) applied Fourier analysis to EEG sequences, which was developed during the 1950s [19].

After that the EEG analysis and classification take grow and development every day. The application of the EEG signals to diagnosis of the brain diseases and to control external devices for disabled people such as wheel chair, prosthesis, etc. Today, several techniques for analysis and classification the EEG signal exists, by using EEG multichannel recording according to 10/20 International electrodes standard, which is used in Brain Computer Interface (BCI).

3 Related works

In this section we present some of related works for EEG data analysis using different techniques such as Non-negative Matrix Factorization (NMF), Normalized Compression Distance (NCD), and Lempel-Ziv (LZ) complexity measure, and Curve Fitting (CF).

Lee et al. presented a Semi-supervised version of NMF (SSNMF) which jointly exploited both (partial) labeled and unlabeled data to extract more discriminative features than the standard NMF. Their experiments on EEG datasets in BCI competition confirm that SSNMF improves clustering as well as classification performance, compared to the standard NMF [10].

Shin et al. have proposed new generative model of a group EEG analysis, based on appropriate kernel assumptions on EEG data. Their proposed model finds common patterns for a specific task class across all subjects as well as individual patterns that capture intra-subject variability. The validity of the proposed method have been tested on the BCI competition EEG dataset [20].

Dohnalek et al. have proposed method for signal pattern matching based on NMF, also they used short-time Fourier transform to preprocess EEG data and

Cosine Similarity Measure to perform query-based classification. This method of creating a BCI capable of real-time pattern recognition in brainwaves using a low cost hardware, with very cost efficient way of solving the problem [5]. In this context, Gajdos et al. implemented the well-performing Common Tensor Discriminant Analysis method [6] using massive parallelism [7].

Mehmood, and Damarla applied kernel Non-negative Matrix Factorization to separate between the human and horse footsteps, and compared KNMF with standard NMF, their result conclude that KNMF work better than standard NMF [14].

Sousa Silva, et al. verified that the Lempel and Ziv complexity measurement of EEG signals using wavelets transforms is independent on the electrode position and dependent on the cognitive tasks and brain activity. Their results show that the complexity measurement is dependent on the changes of the pattern of brain dynamics and not dependent on electrode position [4].

Noshadi et al. have applied Empirical mode decomposition (EMD) and improved Lempel-Ziv (LZ) complexity measure for discrimination of mental tasks, their results reached 92.46% in precision, and also they concluded that EMD-LZ is getting better performance for mental tasks classification than some of other techniques [15].

Li Ling, and Wang Ruiping calculated complexity of sleeping stages of EEG signals, using Lempel-Ziv complexity. Their results showed that nonlinear feature can reflect sleeping stage adequately, and it is useful in automatic recognition of sleep stages [13].

Krishna, et al. proposed an algorithm for classification of the wrist movement in four directions from Magnetoencephalography (MEG) signals. The proposed method includes signal smoothing, design of a class-specific Unique Identifier Signal (UIS) and curve fitting to identify the direction in a given test signal. The method was tested on data set of the BCI competition, and the best result of the prediction accuracy reached to 88.84 % [9].

Klawonn, et al. have applied Curve Fitting for Short Time Series biological data to remove noise from measured data and correct measurement errors or deviations caused by biological variation in terms of a time shift etc. [8]

4 Similarity

The main property in the similarity is a measurement of the distance between two objects. The ideal situation is when this distance is a metric [21]. The distance is formally defined as a function over Cartesian product over set S with non-negative real value (see [3, 12]). The metric is a distance which satisfy three conditions for all:

Definition 1. *A mapping $D : U \to \mathbb{R}^+$ is said to be a distance on the universe U if the following properties hold:*

D1 Non-negativity: $D(x, y) \geq 0$ for any $x, y \in U$;
D2 Symmetry: $D(x, y) = D(y, x)$ for any $x, y \in U$;

D3 Identity of indiscernibles: $D(x, y) = 0$ *if and only if* $x = y$*;*
D4 Triangular inequality: $D(x, y) \leq D(x, z) + D(z, y)$ *for any* $x, y, z \in U$*.*

4.1 Lempel-Ziv Complexity

The Lempel-Ziv (LZ) complexity for sequences of finite length was suggested by Lempel and Ziv [11]. It is a non-parametric, simple-to-calculate measure of complexity in a one-dimensional data. LZ complexity is related to the number of distinct substrings and the rate of their recurrence along the given sequence [17], with larger values corresponding to more complexity in the data. It has been applied to study the brain function, detect ventricular tachycardia, fibrillation and EEG [22]. It has been applied to extract complexity from mutual information time series of EEGs in order to predict response during isoflurane anesthesia with artificial neural networks [2]. LZ complexity analysis is based on a coarse-graining of the measurements, so before calculating the complexity measure $c(n)$, the signal must be transformed into a finite symbol sequence. In this study, we have used turtle graphic for conversion of measured data into finite symbol sequence P. The sequence P is scanned from left to right and the complexity counter $c(n)$ is increased by one unit every time a new subsequence of consecutive characters is encountered. The complexity measure can be estimated using the algorithm described in [11, 2].

In our experiment we do not deal with the measure of the complexity. We create a list of the LZ sequences from the individual subsequence. One list is created for each data file with turtle commands of the compared files.

The comparison of the LZ sequence lists is the main task. The lists are compared to each other. The main property for comparison is the number of common sequences in the lists. This number is represented by the s_c parameter in the following formula, which is a metric of similarity between two turtle commands lists.

$$SM = \frac{s_c}{min(c_1, c_2)} \tag{1}$$

Where

- s_c – count of common string sequences in both dictionaries.
- c_1, c_2 – count of string sequences in dictionary of the first or the second data trial.

The SM value is in the interval between 0 and 1. The two documents are equal if $SM = 1$ and they have the highest difference when the result value of $SM = 0$.

5 Dataset

The data for our experiments was recorded in our laboratory. We have used 7 channels from recorded data. The signal data contains records of the movement of one finger from four different subjects. Every subject performed a press of a button with left index finger. The sampling rate was set to 256 Hz. The signals were band pass filtered from 0.5 Hz to 60 Hz to remove unwanted lower and higher frequencies and noise. The data was then processed, that we extract each movement from the data as well as 0.3s before the movement and 0.3s after the movement.

The pre-processed data contains 4606 data trials – 2303 data trails with finger movement and 2303 trails without finger movement. We divided it into seven groups, one group for each sensor. In our experiment we are using 75% of data for training and 25% for testing. Each group contains part of training and testing data part. The training part for one sensor contains 492 trials – 246 data trails with finger movement and 246 trails without finger movement. The testing part contains 166 trails – 83 trails with finger movement and 83 trails without finger movement. The we have used for further model validation.

5.1 Interpolation of the EEG data

After recording and filtering of the EEG data we apply polynomial curve fitting for data smoothing. The fitting will remove noise from the data and fit the data trend.

Consider the general form for a polynomial fitting curve of order j:

$$f(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \ldots + a_jx^j = \sum_{k=1}^{j} a_kx^k \tag{2}$$

We minimized the total error of polynomial fitting curve with least square approach. The general expression for any error using the least squares approach is:

$$err = \sum (d_j)^2 \tag{3}$$

$$err = (y_1 - f(x_1))^2 + (y_2 - f(x_2))^2 + \ldots + (y_j - f(x_j))^2 \tag{4}$$

$$err = \sum_{i=1}^{n} \left(y_i - \left(a_0 + \sum_{k=1}^{j} a_kx^k \right) \right)^2 \tag{5}$$

where:

- n is count of data points in one move,
- i is the current data point being summed,
- j is the polynomial order.

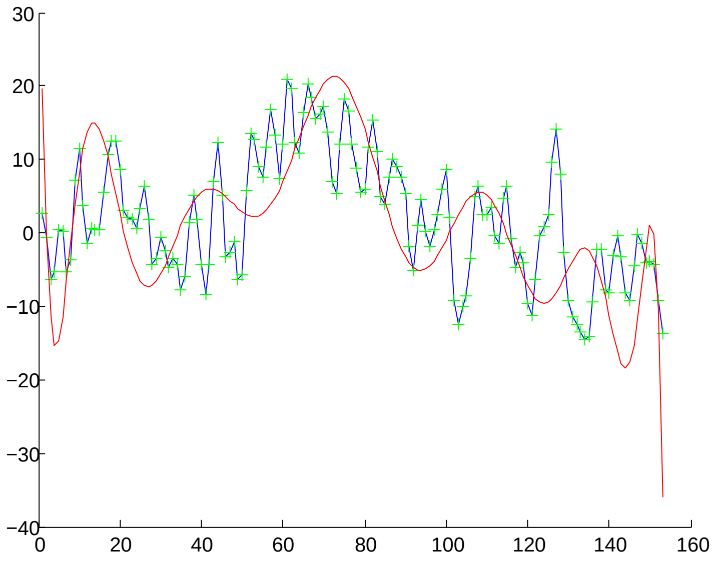

Fig. 2. EGG trail before (blue line) and after smoothing (red line) with 15th order polynomial curve fitting.

5.2 Turtle Graphics

Consider we have control on a turtle on computer screen, this turtle must be respond on a sequence of commands. These commands: forward command, is moving the turtle in front direction a few number of units, right command rotate turtle in clockwise direction a few number of degrees, Back command and Left command are cause same movement but in opposite way. The number of commands to determine, how much to move is called input commands, depending on the application. When moving the turtle under input commands it leave trace, this trace represent the desired object, as in Figure 3. represent simple example for drawing on screen by steering the turtle with four commands forward, right, left, and back command [1]. By this way can represent and drawing the objects, from simple to complex objects.

6 EEG Experiment

The recorded data trail were filtered with band pass filter and divided into individual sensor trails. For each trial we calculated polynomial fitting curve with 15th order and total error minimization with least square approach. The 15th order is enough flexible to smooth data, remove unwanted noise, and to keep the trend of data. After smoothing data we converted calculated curve values into text using Turtle graphics. For the turtle we used 128 commands in two right quadrants – first and fourth. Each command represents one angle – a data

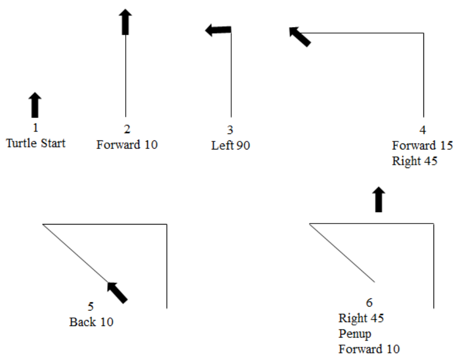

Fig. 3. Simple sequence of Turtle Commands

trend direction. We used only two quadrants – the first and fourth, because the time line goes from left to right and the signal does not go backwards into past.

6.1 Lempel-Ziv Complexity

After this steps, were prepared a LZ subsequences list from turtle graphics commands list from previous step using LZ complexity for each test EEG trial. Similarities to all train trails using Eq. 1 were calculated for every test trail. Then we selected a group of training trails with similarity S satisfying following condition $S \geq T_{min} \land S \leq T_{max}$ for every test trail. The condition threshold values are depicted in Table 1 for all sensors. This selected group of trials is used for calculation in which category belongs the tested trial. This was calculated as a ratio of trials with movement to total count of selected trials in group, using the formula:

$$C = \frac{m_t}{c_t}$$

Where:

- m_t is a count of trails, which are marked as trail with movement,
- c_t is a count of trials in selected group, which satisfy condition.

The tested trail is marked as trail, which belongs to category with movement trails if $C \geq 0.5$ and as a trail without movement otherwise. These steps were performed separately for all categories of data – with movement and without movement – and all sensors.

The values of T_{min} and T_{max} represent the shortest range R in which classifier has correctly identified maximum trials of both categories, with movement and without movement, with emphasis to maximum correctly identified trial with movement, where $T_{min} \in [0, 1]$ and $T_{max} \in [0, 1]$ and $T_{min} < T_{max}$, for example:

$$R(T_{min}, T_{max}) \in [0.15, 0.2]$$

Figure 4 shows a distribution of individual similarities for a trail with (blue bars) and without movement (orange bars). We can see that each data category can be divided into one group. This two groups have with a small intersection between T_{min} and T_{max} value.

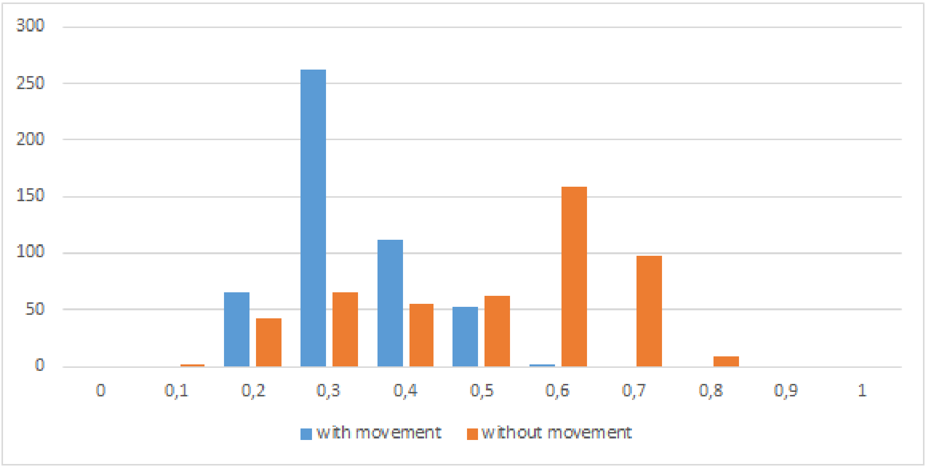

Fig. 4. Histogram of the similarities for trial with and without movement of one sensor

6.2 Experiment Result

Our experiment is focused on successful detection of both data categories, data with movement and data trial without movement. Our data was divided into seven data parts. Each part contains trails from one sensor. Each data parts has two subparts. The first data subpart contains training data – 75% of trials with movement and without movement. The other part is used as testing data sub part. This is used for our model validation. It contains 25% of trials with movement and without movement.

In our experiment we are able to detect movement of index finger with success detection rate between 56.02% and 58.78%. The best results we reach up on sensor S5 (58.78%) and S2, S4 (58.43%). The worst result is for sensor S7 (56.02%). The detection results and their corresponding threshold values for all sensor are in Table 1.

Detection rate in trials with movement varies between 36.14% (S6) and 72.28% (S7). Detection rate in trials with no movement varies between 39.75% (S7) and 77.10% (S6).

Most of the values taken by *minThreshold* are around 0.30 and *maxThreshold* values are situated around value 0.50.

Table 1. Table of Results

Sensor	T_{min}	T_{max}	**Detection rate**		**Total detection rate**
			Movement	**No movement**	
S1	0.40	0.65	61.44%	51.80%	56.62%
S2	0.35	0.45	60.24%	56.62%	58.43%
S3	0.30	0.45	59.03%	55.42%	57.22%
S4	0.30	0.60	66.26%	50.60%	58.43%
S5	0.30	0.40	49.39%	68.29%	58.78%
S6	0.60	0.65	36.14%	77.10%	56.62%
S7	0.25	0.50	72.28%	39.75%	56.02%

7 Conclusion

We made our experiments on our EEG data recorded in our laboratory from four different subjects performing the same task – pressing a button with index finger. The EEG data was recorded using 7 channels recording machine with sampling frequency 256 Hz. The signals were band pass filtered from 0.5 Hz to 60 Hz to remove unwanted frequencies and noise. The signals record the movement of one finger. After removing unwanted frequencies and noise we preprocessed data with polynomial curve fitting with 15th order, turtle graphic – conversion from number into text and Lempel-Ziv complexity – similarity measurement.

In this paper we applied a successful approach for index finger movement detection. Our suggested approach use polynomial fitting curve for smoothing recorded data and Lempel-Ziv complexity for measuring similarity between trails. Our approach is able to correctly detect EEG trail of index finger with success rate between 56.02% and 58.78%. The best results we reach up on sensor 58.78% and 58.43%. The worst result is for sensor 56.02%. Detection rate in trials with movement varies between 36.14% and 72.28%. Detection rate in trials with no movement varies between 39.75% and 77.10% .

The method proposed in this work seems to be able to detect trails with and without movement with overall successful rate more than 56.02%. It can be applied to the use on real data.

Acknowledgment

This work was partially supported by the Grant of SGS No. SP2014/110, VŠB-Technical University of Ostrava, Czech Republic, and was supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070) and by the Bio-Inspired Methods: research, development and knowledge transfer project, reg. no. CZ.1.07/2.3.00/20.0073 funded by Operational Programme Education for Competitiveness, co-financed by ESF and state budget of the Czech Republic.

References

1. H. Abelson and A. diSessa. *Turtle Geometry: The Computer as a Medium for Exploring Mathematics.* The MIT Press, July 1986.
2. D. Abásolo, R. Hornero, C. Gómez, M. García, and M. López. Analysis of EEG background activity in alzheimer's disease patients with lempel–ziv complexity and central tendency measure. *Medical Engineering & Physics*, 28(4):315 – 322, 2006.
3. R. Cilibrasi and P. M. B. Vitányi. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545, 2005.
4. A. de Sousa Silva, A. Arce, A. Tech, and E. Costa. Quantifying electrode position effects in eeg data with lempel-ziv complexity. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 4002–4005, 2010.
5. P. Dohnálek, P. Gajdoš, T. Peterek, and M. Penhaker. Pattern recognition in EEG cognitive signals accelerated by GPU. volume 189 AISC, pages 477–485. 2013. cited By (since 1996)1.
6. A. Frolov, D. Husek, and P. Bobrov. Brain-computer interface: Common tensor discriminant analysis classifier evaluation. In *Nature and Biologically Inspired Computing (NaBIC), 2011 Third World Congress on*, pages 614–620, 2011.
7. P. Gajdos, P. Dohnalek, and P. Bobrov. Common tensor discriminant analysis for human brainwave recognition accelerated by massive parallelism. In *Nature and Biologically Inspired Computing (NaBIC), 2013 World Congress on*, pages 189–193, 2013.
8. F. Klawonn, N. Abidi, E. Berger, and L. Jänsch. Curve fitting for short time series data from high throughput experiments with correction for biological variation. In J. Hollmén, F. Klawonn, and A. Tucker, editors, *IDA*, volume 7619 of *Lecture Notes in Computer Science*, pages 150–160. Springer, 2012.
9. S. Krishna, K. Vinay, and K. B. Raja. Efficient meg signal decoding of direction in wrist movement using curve fitting (emdc). In *Image Information Processing (ICIIP), 2011 International Conference on*, pages 1–6, 2011.
10. H. Lee, J. Yoo, and S. Choi. Semi-supervised nonnegative matrix factorization. *Signal Processing Letters, IEEE*, 17(1):4–7, 2010.
11. A. Lempel and J. Ziv. On the complexity of finite sequences. *Information Theory, IEEE Transactions on*, 22(1):75–81, 1976.
12. M. Li, X. Chen, X. Li, B. Ma, and P. M. B. Vitányi. The similarity metric. *IEEE Transactions on Information Theory*, 50(12):3250–3264, 2004.
13. L. Ling and W. Ruiping. Complexity analysis of sleep eeg signal. In *Bioinformatics and Biomedical Engineering (iCBBE), 2010 4th International Conference on*, pages 1–3, 2010.
14. A. Mehmood and T. Damarla. Kernel non-negative matrix factorization for seismic signature separation. *Journal of Pattern Recognition Research*, 8(1):13–25, 2013.
15. S. Noshadi, V. Abootalebi, and M. Sadeghi. A new method based on emd and lz complexity algorithms for discrimination of mental tasks. In *Biomedical Engineering (ICBME), 2011 18th Iranian Conference of*, pages 115–118, 2011.
16. R. Q. Quiroga. *Quantitative analysis of EEG signals: Time-Frequency methods and Chaos Theory.* PhD thesis, Institute of Signal Processing and Institute of Physiology, Medical University of Lubeck, Germany, 1998.
17. N. Radhakrishnan and B. Gangadhar. Estimating regularity in epileptic seizure time-series data. *Engineering in Medicine and Biology Magazine, IEEE*, 17(3):89–94, 1998.

18. T. K. Rao, M. R. Lakshmi, and T. V. Prasad. An exploration on brain computer interface and its recent trends. *International Journal of Advanced Research in Artificial Intelligence*, 1(8):17 – 22, 2012.
19. S. Sanei and J. Chambers. *EEG Signal Processing*. John Wiley & Sons Ltd., The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England, 2007.
20. B. Shin and A. Oh. Bayesian group nonnegative matrix factorization for eeg analysis. *CoRR*, abs/1212.4347:1–8, 2012.
21. A. Tversky. Features of similarity. *Psychological Review*, 84(4):327–352, 1977. cited By (since 1996)1968.
22. X.-S. Zhang, R. Roy, and E. Jensen. Eeg complexity as a measure of depth of anesthesia for patients. *IEEE Transactions on Biomedical Engineering*, 48(12):1424–1433, 2001. cited By (since 1996)165.

Exploiting HTML5 Technologies for Distributed Parasitic Web Storge*

Martin Kruliš, Zbyněk Falt, Filip Zavoral

Parallel Architectures/Applications/Algorithms Research Group
Faculty of Mathematics and Physics, Charles University in Prague
Malostranské nám. 25, Prague, Czech Republic
{krulis,falt,zavoral}@ksi.mff.cuni.cz

Abstract. Current web technologies have been leaping forward, especially since the introduction of HTML5. The web browsers of the day implement various APIs for the client-side scripts, such as elaborate data storage or advanced network connectivity. We propose, how to combine these technologies to create distributed data storage using the web environment. We have implemented a prototype framework as a proof of concept and explored the most problematic issues which require to be researched further. Systems that would use this storage may benefit from the fact that the users do not need to install or configure any type of client application, since the only thing required is the web browser. Web-based distributed data storage can be used as an extension of server storage, for data caching, and in various other applications.

Key words: html5,web storage,distributed,data,parasitic

1 Introduction

World wide web played a role of open interactive platform for information exchange, business, or entertainment for over twenty years. It begun as a simple technology for presenting text documents, but it has evolved significantly in the past decades. In the last few years, its technologies leaped forward as HTML5 [1] emerged, which is not only a new version of the HTML language, but it also integrates specifications for client side scripting and APIs for advanced browser features. It shifted the web browsers from simple web page presenters into an application platform, which even become basis for lightweight operating systems [2].

The ability of executing scripts in the browser raises many security issues, since the user have virtually no way how to verify that these scripts do not perform malicious routines and will not pose a threat. The client scripting environments in the browsers must find a very fine balance between security and provided functionality, which is required by complex client-server applications. Furthermore, the browsers must implement these specifications correctly to ensure at least some level of protection.

* This paper was supported by Czech Science Foundation (GAČR) projects P103/14/14292P and P103/13/08195 and by specific research SVV-2014-260100.

The HTML5 technologies are designed to support more complex client-side scripts for more elaborated client-server web applications. Particularly, HTML5 advanced these browsers capabilities:

- GUI design, graphics, and multimedia support,
- communication capabilities (Server-sent events, WebSockets, WebRTC),
- client-side data and file management (File API, Web Storage, Indexed DB),
- client-side computations (Web Workers, WebCL).

The line between legitimate resource utilization and their exploitation is quite thin and not formally defined. Therefore, it is reasonable to assume that these technologies will remain in the browsers for the perceivable future time, despite the possibility they might be misused by parasitic applications.

1.1 Contributions and Outline

Our particular interest turns towards data storage capabilities and communication capabilities of the browsers. Many current users of the web have a lot of free disk space in their computers and sufficient connectivity. This space may be utilized to store or cache data of the web applications they use. It may also be exploited by a web application, or its embedded component such as ad banner, to secretly infest the users hard drive with their own data.

This paper explores the limitations of current HTML5 technologies and propose a way how these technologies may be used or misused to create a distributed data storage. We have implemented a prototype framework that tests the feasibility of this idea. The prototype let us identify the most problematic areas of this domain and outline possible solutions for them. We have also proposed possible applications which may use these technologies for legitimate reasons and in a parasitic way – i.e., stealing the disk space from the users.

The paper is organized as follows. Section 2 revises related work. The HTML5 technologies relevant to our work are presented in Section 3. Section 4 proposes a design of distributed data storage and outlines possible problems that will require further research. Possible applications are presented in Section 5 and Section 6 concludes the paper.

2 Related Work

Utilizing unused resources from desktop computers is not a new idea. Many systems, such as Entropia [3] or SETI@home [4], have successfully used idle CPU in the past. With new HTML5 technologies, this task become even easier. For instance a WeevilScout framework [5] was built to utilize computational power of the web browsers [6] for bioinformatic tasks.

In this work, we focus on utilizing the spare storage capacities of ordinary computers to form a distributed data storage. Distributed storage systems of the day have evolved from providing basic means of remote data, to offering services like high availability, anonymity, redundancy, and archival [7, 8].

Several projects already looked into scavenging unused storage on desktop computers. Farsite [9] is a replication-based serverless distributed file system built on Windows desktops that uses the Byzatine-fault protocol to manage file and directory metadata. Freeloader [10] framework aggregates unused desktop storage space into a shared cache space for hosting large scientific datasets. It utilizes file stripping across a set of machines in order to provide high data throughput. Storage@home [11] is a distributed storage infrastructure developed to solve the problem of backing up and sharing large data using a distributed model of volunteer managed hosts.

As a peer-to-peer storage system, CFS [12] stores data blocks reliably using distributed hash tables. It arranges blocks over a number of nodes to provide fault tolerance, high scalability, and load balancing. Ivy [13] is a file system using logs to saving both data and meta-data. The logs are saved in the distributed hash table across the unreliable peers. The Freenet [14] file sharing network stores anonymized documents and allows them to be retrieved later by an associated key. Files on Freenet are split into multiple small blocks, with additional blocks added to provide redundancy. Also peer-to-peer file sharing networks such as BitTorrent [15] or DC++ [16] can be considered as representants of such a class of distributed storage systems. Using decentralized network, the users share their files with other members.

Although all these storage systems proved their usability and applicability, they suffer from one substantial drawback from a user's point of view: they require to install a client software. To our best knowledge, no distributed storage system operates without the need of such additional client component.

3 HTML5 and New Web Technologies

In this section, we will briefly introduce the HTML5 technologies, which have emerged in the past few years and which may be used to create distributed data storage using web browsers of common users. These technologies are implemented in the leading desktop browsers, thus available for most web users.

3.1 Persistent Data Storage

The HTTP was designed as strictly stateless protocol, with no user session support. This flaw was partially remedied by introducing *cookies* – a simple mechanism, that allows server to register session data at client side, which are resent automatically with each request. Cookies do not satisfy complex needs of modern web applications, thus more elaborate solutions were devised in HTML5.

Web Storage. The web storage [17] was originally designed with the same motivation as cookies – to store structured data at client side. Unlike cookies, the web storage is completely maintained by JavaScript, so the programmer may use it in different ways. There are two types of web storage interfaces for two different scenarios – the session storage and the local storage.

The *session storage* is designed for scenarios when the application needs to store session data, but the user may run multiple sessions in different windows. Therefore, the storage is tied to a specific site and specific opened window/tab.

The *local storage* is designed for scenarios when the application needs to store data at the client side, but these data span throughout the opened windows. It is also particularly useful when the data needs to overlive the current user session. The storage is tied to a specific site and the data are persistent.

Both storages use simple API that allows saving key-value pairs, where both the key and the value are simple strings. The keys are treated as unique identifiers and the API allows the application to iterate over all stored keys. The specification also includes some security restriction. The most important restriction in our case is the disk quotas for the stored data, i.e., the browser does not permit the script to store more data then specified by the quota.

Indexed Database. More complex data management problems require more elaborate solutions. For this purpose, the W3 consortium presented the Index Database API [18], which provide basic database functionality for the client scripts. The IndexDB API replaces Web SQL Database API, which was deprecated by W3C, yet some implementations exist.

The indexed database API allows the JavaScript at client side to create indexed data stores and operate them in a transaction-safe way. The data are usually stored in efficient persistent data structures (such as B-trees), which support efficient insertion, deletion, key-based searching, and deterministic (ordered) enumeration.

3.2 Communication Capabilities

The first direct communication capabilities were introduced into client-side web scripting with the asynchronous HTTP API (`XMLHttpRequest`). This API allowed the client code to perform its own requests on the background, thus synchronize client and server application status without explicit user actions. However, the communication must still be initiated from the client side, which complicates situations when the server needs to notify clients.

One of the first techniques designed for server-initiated notifications (also denoted AJAX-push or reverse AJAX) was Comet [19]. This technique uses long-held HTTP requests that allow the server to initiate the transfer over a connection created by the client. The client sends an asynchronous HTTP request to the server demanding a status update. If no updates are available, the server postpones the response whilst the connection remains open. When the status changes, the server notifies all clients by sending the responses to all pending requests. Unfortunately, the long-held HTTP requests are not managed very well by current HTTP servers (e.g., Apache or Microsoft IIS).

An extension of the Comet approach was presented by the Server-sent events API. This API uses also long-held HTTP connections, but it allows multiple events to be sent over an opened HTTP request and standardizes both the message format and the client-side API that process these messages.

WebSocket. The WebSocket protocol is a HTTP-compatible protocol, which allows upgrading regular HTTP connections into a WebSocket connections. The WebSocket connection reuses underlying TCP (or SSL/TLS) channel of the original connection to transfer data frames in either direction. Both sides (former HTTP client and server) become equal communication partners, i.e., they can send a message over to their peer at any time.

The WebSocket client API [20] is quite simple. It handles only the creation and the destruction of the communication channel and sending/receiving messages. The protocol supports both textual and binary data, thus the JavaScript may send or receive strings, `Blob` objects, and `ArrayBuffer` objects.

WebRTC. The HTTP protocol and the WebSockets are designed for client-server communication. In some cases, a direct communication between clients may be required. WebRTC [21] is a technology, which was designed for real-time peer-to-peer communication between web browsers. The technology was originally intended for multimedial transfers (e.g., video phone calls), but it handles data transfers as well.

The browsers first create a `RTCPeerConnection`, which is a logical representation of the connection. This connection is usually routed using ICE technologies for NAT traversal, since most browsers are connected to the internet from a private network. The RTC connection allows the transfer of one or multiple streams. A stream may be either media stream, which can be directly captured from a web camera and directly displayed in an appropriate HTML5 element, or a data stream, which has the same behaviour and API as a WebSocket connection.

4 Web Browser as Distributed Storage Platform

A distributed data storage system (as almost any database system) usually employs layered architecture. Lower layers are responsible for raw data operations while upper layers add user-friendly functionality, such as transparent format transcoding, management of logical data entities, integrity constraints, transaction support, or security. These high-level features are quite well understood and beyond the scope of this work. Hence, we focus on the lowest level – the utilization of HTML5 technologies for raw data storage and transfer.

4.1 Infrastructure

The data storage infrastructure can be divided into three logical components:

- the master (running mostly on a server or on multiple servers),
- the clients (the connected web browsers),
- and the communication protocol they use.

The master controls the entire storage. It integrates all data distribution logic and initiates all data transactions. It can be implemented as one centralized server, as a small cluster of servers located in one server room, or even as a large distributed system. Theoretically, some parts of the master functionality can be performed in the browsers; however, these details are not important from the data storage point of view. Hence, we will present the master as a single server.

The most important role of *the clients* is to provide their storage space for the system. Some of these clients may also use the system functions (i.e., retrieve/store user data); however, we primarily focus on the data management. The clients require to execute only a minimalist script which connects to the master and waits for commands. All operations are issued from the master and the client simply process them. There are three basic kinds of operations:

- metadata operations (client identification, capabilities assessment),
- data retrieval (list, read),
- and data modifications (insert, update, delete).

The protocol is designed to work as a simple RPC for the client operations. The data transfers are bidirectional and server-initiated, thus we selected the WebSockets as the underlying protocol. Each operation has its own request message and response message. The client process the messages sequentially and for each incoming request generates one response. Figure 1 illustrates this model.

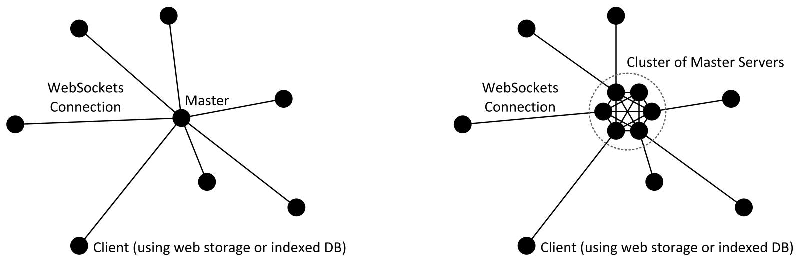
WebSockets Connection
Master
Client (using web storage or indexed DB)
Cluster of Master Servers
WebSockets Connection
Client (using web storage or indexed DB)

Fig. 1. Data storage with centralized data transfers

Optimizing Data Transfers. Even though the basic concept may suggest that the major data flow will be between master and clients, there are many situations, when the data needs to be transferred between clients directly. For instance, when a data block needs to be replicated from one nodes to another or when a client requests data, which are stored on adjacent nodes.

The WebRTC peer-to-peer connections may be used to optimize data transfers between the clients (browsers). The WebRTC creates data streams, which are operated by the same API as the WebSockets. Hence, it will be easy to adapt the implementation to accomodate this feature. Even though the WebRTC may reduce the master-client communication, the uplink to the master is still required, since the master controls the performed operations.

Figure 2 depicts the augmentation of centralized data storage where peer-to-peer data channels are used to optimize the data traffic between clients. The peer-to-peer channels are created on demand only between those clients that need to communicate and they may be closed when no longer needed.

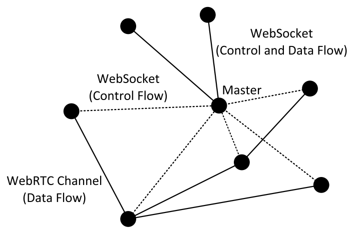

Fig. 2. Distributed data storage with peer-to-peer channels

4.2 Practical Observations

We have implemented the prototype of the data storage layer. The client was implemented as a simple web page which connects to the master using WebSocket [20] channel and handle commands sent by the server. Both web storage [17] and Indexed DB API [18] were tested for the client-side data management. The master was implemented as Node.js application that distributed data to the connected clients and verified the accessibility of the data.

Addressing all technical details is way beyond the scope of this work, but we have identified the most problematic issues and proposed solutions for them.

Resource Limitations. One of the key issues is the the limitation of user resources, especially the disk space. Current browser use the default quota of 5 MB per site for local storage [17]. This limit is to low to create large distributed data storage even if millions of users are participating. The quota can be configured in all major browsers, however, changing this setting is not very convenient. The IndexedDB API [18] is designed for significantly larger data, but it still apply some form of quotas. Fortunately, the IndexedDB apply only soft quotas and the browser interactively prompts the user when the limit is to be exceeded. Furthermore, a Quota API [22] is currently being specified by the W3 consortium, which should provide even more convenient way to manage the quotas.

Another resource that might limit the system is the network connectivity. The desktop computers are usually connected by some form of wired connection, such as xDSL (phone lines) or FTTx (dedicated metallic or optical lines). According to various resources, the average connection speeds are oscillating between 1 and 100 MBps depending on the location. Furthermore, we have to consider that many technologies (like the ADSL) use asymmetric connections, where the upload is several times slower than the download.

The storage and the data transfer issues are even more serious if we consider portable devices such as tablets or smartphones. Regular user hardly notices if we allocate one gigabyte on a desktop computer, but the same amount of data cannot be easily taken from a smartphone. Furthermore, mobile devices are usually connected via cellular networks which are much slower and often limit the data transfers by fair user policies.

Volatility of the Clients. Most of the distributed systems expect that the participating nodes are devoting their efforts to fulfill the objectives of the system. Although there are methods of dealing with many forms of failures, the system usually expects that the nodes are available most of the time. The web users, on the other hand, connect to the system at their own discretion without any regards to the system requirements.

The system must employ specific measures to ensure data availability. Prehaps the most straightforward solution is to select sufficient data replication level. More elaborate approach would be to study individual behaviour of the users and data requests. These strategies will require intensive future research.

Security. The system must provide some guarantees regarding the stored data, such as persistence, integrity, or privacy. Our distributed data store does not fulfill any of these requirements, since the clients do not provide any guarantees. The data may be easily corrupted, erased, or accessed by unauthorized users. Some of the security requirements may be fulfilled by employing check sums, security hashes, cryptography, and data replication. However, this topic is too broad and beyond the scope of this paper.

5 Web Storage Applications

We have projected initial estimates concerning the web storage capacity based on existing web application and outlined a few possible use cases to illustrate the applicability of the storage.

5.1 Capacity Analysis

We have based our analysis on the statistical data of Facebook [23], which is probably the largest web application as it has the most active users. At the beginning of year 2014, the Facebook had 1.31 billion active users, from which 51.9% used mobile or handheld devices to access the application. The number of users is expected to grow in the future as it has in the past (e.g., the annual growth between 2012-2013 was 22%). If we utilize 1 GB on average of each active desktop user, we could get approximately 600 PB of raw capacity.

Utilizing this raw capacity effectively will be quite a challenging task. Even the users of such popular application as the Facebook exhibit very irregular behaviour in visiting the web page. The active users spend about 640 million minutes each month, which is only one minute per user in average. On the other hand, 48% of all users connect to the web page on a daily basis.

These statistics suggest that the matter of data replication cannot be solved by a simple random approach, but careful study of user behaviour has to be conducted. Furthermore, the application may employ additional methods like benefits to encourage the users connect and stay on the web page.

5.2 Legitimate Usage

The data storage and sharing systems usually require some form of a client application. If such systems are built on the top of web technologies, they may provide clients which can be directly used in the web browser without explicit installation or configuration.

Existing applications, in which the users share some content, can use the distributed storage as a cache to reduce the data transfers of their internal servers. For instance, Facebook can use this cache for photographs, Google Doc for documents, etc. When the data are requested by a user, they may be downloaded from a peer who is online instead of downloading them from a server.

A different approach can be used by applications which have many users, but do not require storing or caching large amounts of data. These applications may offer the capacity of the distributed storage to third parties. The concept is similar to web advertisement where a web page shows ad banners, however in this case, the users support the application by donating their own disk space.

5.3 Parasitic Usage

The data storage can be also misused by malicious applications to steal the disk space from the users. For instance, the idea of acquiring the data storage at clients by the means of ad banners can be used both legitimately and parasitically, depending on whether the user is informed about the intent or not. However, when used parasitically, the application needs to use quite elaborate way of user targeting in order to create at least somewhat persistent data storage.

To acquire even more clients, the parasitic storage malware may resort to exploit vulnerabilities of other web applications. When a victim application is not correctly protected against script injection attacks [24], the parasitic application may inject the client code. The users of the victim application then become also clients of the parasitic distributed storage without realizing it.

6 Conclusions

We have proposed a novel idea how existing HTML5 technologies may be used or even misused for a distributed data storage. The storage clients require only modern web browser, which is a piece of software that is present on virtually every desktop computer. We have implemented prototype framework as a proof of concept and outlined additional problems that require our attention.

In the future work, we would like to address the remaining problem, especially analyze the user behaviour. If we are able to predict client up times based on behavioural models, we will be able to distribute data among the client nodes in more efficient way. Furthermore, we would like to explore possibilities of creating distributed database system, that will also distribute the query evaluation plans and execute them partially on the client nodes.

References

1. Hickson, I., Hyatt, D.: Html5. W3C Working Draft, May (2011)
2. Pichai, S., Upson, L.: Introducing the Google Chrome OS. The Official Google Blog (2009)
3. Chien, A., Calder, B., Elbert, S., Bhatia, K.: Entropia: architecture and performance of an enterprise desktop grid system. Journal of Parallel Distributed Computing **65** (2003) 597–610
4. Univ. of Berkeley: SETI@Home. http://setiathome.ssl.berkeley.edu/ (2006)
5. Reginald, C., Putra, G., Belloum, A., Koulouzis, S., Bubak, M., de Laat, C.: Distributed Computing on an Ensemble of Browsers. (2013)
6. W3C: Web Workers. http://www.w3.org/TR/workers/
7. Thanh, T., Mohan, S., Choi, E., Kim, P.: A taxonomy and survey on distributed file systems. NCM 08, Fourth International Conference on Networked Computing and Advanced Information Management (2008)
8. Hasan, R., Anwar, Z., Yurcik, W., Brumbaugh, L., Campbell, R.: A survey of peer-to-peer storage techniques for distributed file systems. International Conference on Information Technology: Coding and Computing, ITCC 2005 **2** (2005)
9. Adya, A., Bolosky, W., Castro, M., Cermak, G., Chaiken, R., Douceur, J., Howell, J., Lorch, J., Theimer, M., Wattenhofer, R.: Farsite: Federated, available, and reliable storage for an incompletely trusted environment. Proceedings of the 5th Symposium on Operating Systems Design and Implementation (2002)
10. Ma, X., Vazhkudai, S.S., Zhang, Z.: Improving data availability for better access performance: A study on caching scientific data on distributed desktop workstations. Journal of Grid Computing (2009)
11. Beberg, A.L., Pande, V.S.: Storage@home: Petascale Distributed Storage. Parallel and Distributed Processing Symposium (2007)
12. Dabek, F., Kaashoek, M., Karger, D., Morris, R., Stoica, I.: Wide-area cooperative storage with cfs. Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP 01) (2001)
13. Muthitacharoen, A., Morris, R., Gil, T., Chen, B.: Ivy: A read/write peer-to-peer file system. Proceedings of 5th Symposium on Operating Systems Design and Implementation (2002)
14. Clarke, I., Sandberg, O., Wiley, B., Hong, T.W.: Freenet: A Distributed Anonymous Information Storage and Retrieval System. Designing Privacy Enhancing Technologies. Lecture Notes in Computer Science (2001)
15. Pouwelse, J.e.a.: The bittorrent p2p file-sharing system: Measurements and analysis. Peer-to-Peer Systems (2008) 205–216
16. : DC++. http://dcplusplus.sourceforge.net/ (2013)
17. W3C: Web Storage. http://www.w3.org/TR/webstorage/
18. W3C: Indexed Database API. http://www.w3.org/TR/IndexedDB/
19. McCarthy, P., Crane, D.: Comet and Reverse Ajax: The Next-Generation Ajax 2.0. Apress (2008)
20. W3C: The WebSocket API. http://dev.w3.org/html5/websockets/
21. W3C: WebRTC 1.0: Real-time Communication Between Browsers. http://dev.w3.org/2011/webrtc/editor/webrtc.html
22. W3C: Quota Management API. https://dvcs.w3.org/hg/quota/raw-file/tip/Overview.html
23. Statistic Brain Research Institute: Facebook Statistics from 1.1.2014. http://www.statisticbrain.com/facebook-statistics/
24. Spett, K.: Cross-site scripting. SPI Labs (2005)

An Application of Process Mining by Sequence Alignment Methods to the SAP Invoice Process Example

Jakub Štolfa[1], Svatopluk Štolfa[1], Kateřina Slaninová[1,2], Jan Martinovič[1,2]

[1] Department of Computer Science, FEI, VŠB - Technical University of Ostrava,
17. listopadu 15, 708 33, Ostrava-Poruba, Czech Republic
[2] IT4Innovations, VŠB - Technical University of Ostrava,
17. listopadu 15, 708 33, Ostrava-Poruba, Czech Republic
{jakub.stolfa, svatopluk.stolfa, katerina.slaninova,
jan.martinovic}@vsb.cz

Abstract. Process mining started to be a very useful tool how to check processes in companies. The only thing that has to be done is to have at least some log about the activities performed by the software system. There are many methods that can be used then to analyze this data obtained from the log. Every usage of yet not used or rarely used method opens up new perspectives in process mining and reveals the unknown potential of its application to the practice. In this paper we have applied sequence alignment methods to the real process example and examined what results and benefits could be obtained from such usage. The main purpose of this paper is to adjust methods for sequence alignment to be able to determine similarity between the business processes.

Keywords: Sequence Alignment, Process Mining, SAP

1 Introduction

Generally, information systems support business processes. Enactments of the processes are partly managed by the systems, partly managed by users decisions and activities. It is not easy to understand whether the specific process runs efficiently, because usually various activities are processed in parallel and process definition allows plenty of process enactment variations. Our task was to analyze specific process with request to suggest steps for its simplification, curtailment and enact the process cheaper.

The research described in this paper is based on our previous work that included process reconstruction and path analysis [15]. According to this previous research we were able to adjust the process, recognize the false usage of the process, analyze malfunctions in the reality etc. Other previous research closely related to this paper has been done It was focused on the analysis of the process data that involved usage of the sequence alignment methods [9]. The aim of this paper is focused on adjusting of our approach presented in mentioned previous work to the business process area with the consideration to its specific characteristics, especially to adjust methods for sequence alignment to be able to determine similarity between the business processes. The approach was tested and used in other research areas yet, for example in e-learning area [12], in analysis of behavior of agents during the simulation [14] and in analysis of user behavior on the web [13].

The paper is organized as follows: Section 2 introduces the state of the art; Section 3 describes the process that is analyzed and log obtained from the company, Section 4 depicts the experiment that we have performed, describes the preparation of data that we have obtained, shows the usage of our process mining method and explains obtained results; concluding Section 5 provides a summary and discusses the planned future research.

2 State of the art

Business process definitions are sometimes quite complex and allow many variations. All of these variations are then implemented to supportive systems. If you want to follow some business process in a system, you have many decisions and process is sometimes lost in variations. Modeling and simulations can help you to adjust the process, find weaknesses and bottlenecks during the design phase of the process.

The idea of process mining was introduced by Aalst in 2004 [1, 2]. This area of the research has been developing during the years, lot of methods were introduced to this topic. In 2005, ProM tool was introduces [3]. ProM aggregates methods and approaches in this area of study. There are a lot of papers that describe new ways or improvements of methods, techniques and algorithms used in the process mining, but only several papers are focused on the case studies [8].

In the area of process mining the methods of the sequence alignment were introduced by Esign and Karagoz [4] in 2013. Focus of their work was quantitative approach for performing process diagnostics. The approach uses sequence alignment methods for delta analysis. It is comparison of actually performed process and prescriptive reference model [1]. Our paper provides another usage of the sequence alignment methods. We use these methods for comparison of extracted processes to find similarity in the process executions, i.e. some patterns of the process.

The basic approach to the comparison of two sequences, where the order of elements is important, is The longest common substring method (LCS). This is used in exact matching problems [6]. It is obvious from the name of the method that its main principle is to find the length of the common longest substring. The LCS method respects the order of elements within a sequence. However, the main disadvantage of this method is that it can only find the identical subsequences, which meet the characteristics of substrings.

Unlike substrings, the objects in a subsequence might be intermingled with other objects that are not in the sequence. The longest common subsequence method (LCSS) allows us to find the common subsequence [7]. Contrary to the LCS method, the LCSS method allows (or ignores) these extra elements in the sequence and, therefore, it is immune to slight distortions.

The third selected method was The time-warped longest common subsequence (TWLCS) [5]. This method combines the advantages of the LCSS method with dynamic time warping [10]. Dynamic time warping is used for finding the optimal visualization of elements in two sequences to match them as much as possible. This method is immune to minor distortions and to time non-linearity. It is able to compare sequences, which are for standard metrics, evidently not comparable.

The methods LCS and LCSS used for the comparison of sequences find the longest common subsequence z of compared sequences x and y, where $(z \subseteq x) \wedge (z \subseteq y)$. The relation weight $w_{seq}(x, y)$ between the sequences x and y was counted by Equation 1:

$$w_{seq}(x, y) = \frac{l(z)^2}{l(x)l(y)} \frac{Min(l(x), l(y))^2}{Max(l(x), l(y))^2}, \tag{1}$$

where $l(x)$ and $l(y)$ are lengths of the compared sequences x and y, and $l(z)$ is a length of a subsequence z. Equation 1 takes account of the possible difference between $l(x)$ and $l(y)$. Due to this reason, z is adapted so that $w_{seq}(x, y)$ is strengthened in the case of similar lengths of sequences x and y, and analogically weakened in the case of higher difference of $l(x)$ and $l(y)$. For the methods LCS and LCSS, w_{seq} meets all the similarity conditions: $w_{seq} \geq 0$, $w_{seq}(x, x) = 1$, $w_{seq}(x, x) > w_{seq}(x, y)$ and $w_{seq}(x, y) = w_{seq}(y, x)$.

The output z is only the sequence which characterizes the relation between the sequences x and y for T-WLCS method. Therefore, $w_{seq}(x, y)$ does not meet all the similarity conditions due to its characteristics. Respectively, it is possible that $w_{seq}(x, y) > w_{seq}(x, x)$. Although we know that $w_{seq}(x, y)$ is not a similarity for T-WLCS method, due to a simplification, the 'sequence similarity' will be used as a relation weight $w_{seq}(x, y)$ between the sequences x and y for all the methods of sequence comparison in the following text.

As a complementary method for comparison of sequences we have used common method used in informational retrieval, Leventhtein distance [11].

3 Process Context

The analyzed company runs SAP system in five countries and process approx. 30,000 supplier invoices per year. Examined business process of the invoice verification is implemented in SAP ERP and SAP DMS, user activities are controlled by SAP business workflow. Users participate in the invoice verification workflow in several different roles (creator, accountant completion, approver, and accountant decision and posting). Generally, it is process where the accountant should create the invoice, verify it, send to the approvers and finally when he gets it back he does invoice posting.

We have loaded the log of the process between 1/1/2012 and 6/30/2012, totally we loaded 37,991 records for adjusting. Detailed description of the obtaining log and data preprocessing is described in our previous work [15].

4 An Application of the Sequence Alignment

The main purpose of this paper is to adjust methods for sequence alignment to be able to determine similarity between the business processes. The first step of the experiments was focused on one characteristic of the business processes: the duration of the events. In sequence alignment area, this problem is not common. That is the main reason for adjustment of the methods to this area. We have performed an experiment where we made decision about the categorization of the duration of events, prepared four types of sequences from the different viewpoints, analyzed the prepared sequences and applied the sequence alignment methods to determine the similarities of the sequences. Finally, we have selected the right method for our purposes and analyzed the applicability of our approach on the example. The following subsections describe the particular steps in detail.

4.1 Data Preparation and Categorization - Duration Category

When we look at the histogram of all events that is depictured at Fig. 1, we can see that there are cases that where completed in one day, then cases that last two days etc. We can see interesting distribution of the process duration on this histogram that looks like the waves. The waves are caused by the fact that all events of this process are performed in window of approximately 8 working hours each day. Then there is a 16 hours delay till the next working day window appears.

We needed to categorize the duration of the events. After the consultation with the company, we have decided to set three categories for the event duration. First category is the process that last up to the 168 hours. It means that they were started in the window of one working week and ended the same week or the week after. Next, the second category is the category of processes that last from one week to one month. The last category is the category of all processes that last more than one month. Since there are four events in the process, one process should last up to 36 hours so all four together last up to one week. Similar categorization principle works for other categories too.

We set up aliases for the type of the events in the sequence. It means that Verification event is in the sequence like V, Creation event is C, Approval is A, and Posting is P.

4.2 Types of Event Sequences

According to information from the data log we could analyze information in detail and from different points of view. We have defined four types of points of view, or we can say type of event cases:

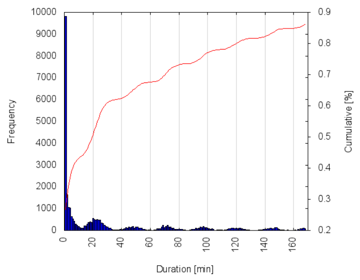

Fig. 1. Histogram of all events

– Type A - Events without Time and Users,
– Type B - Events with Time and without Users,
– Type C - Events without Time and with Users,
– Type D - Events with Time and Users.

Case type A focuses on the topological structure of the process only and does not care about meta-information of the events like time, or duration of the event, or user that performs particular event.

Example of the sequence for the case type A: `0,C;V;A;A;S;`, where sequence is defined by the following structure `CaseID, event1;event2;event3;...;eventn`

Case type B focuses on the topological structure of the process and combines it with the information about time, or duration, of the events. That can bring us another point of view to the process. Thanks to this we can see duration of the whole process, or its events. Tagging of the duration of the events is made by repeating the symbol of the event in the sequence. How much is the symbol repeated depends on the duration category of the particular event. If the event fits in the category three than the symbol is repeated three times, if the event fits in the category two than the symbol is repeated two times and for the category one is symbol repeated one time.

Example of the sequence for the case type B: `0,C;C;V;A;A;S;`, where sequence is defined by the following structure: `CaseID, event1; event1 (repetition according to the duration cat.); event2; ...; eventn`.

Case type C combines topological structure of the process and metainformation about the users. It brings us interesting view to the users involved in particular process.

Example of the sequence for the case type C: `0,C_USER068; V_USER068; A_USER272; S_USER068;`, where sequence is defined by the following structure: `CaseID, event1_originator of the event1; event2_originator of the event2; event3_originator of the event3;...; eventn_originator of the eventn`.

Case type D combines all possible views to the process - topological, time view and users view. It can bring us superb and surprising view to the process that is not possible to see at the first glance.

Example of the sequence for the case type D: `0,C_USER260; C_USER260;V_USER260;A_USER074;A_USER202;S_USER260;`, where sequence is defined by the following structure: `CaseID, event1_originator of the event1(repetition according to the duration category); event2_originator of the event2; event3_originator of the event3;...; eventn_originator of the eventn`.

Thanks to the different case types we can discover processes that take least time, most time to accomplish it, or we can find deviations in the process execution that is not possible to see only in the topological view. We can see if some users take more time to execute event, or if some users communicate more or less, etc.

4.3 Information about sequence types sequence distribution

We have made sequence distribution diagrams for four case types. Some basic interesting information about particular process can be identified here.

Fig. 2 on the left side shows histogram of the most frequented sequences. Histogram nicely shows distribution of the frequency of the particular sequences. It is focused to the case type A. We can see that most frequented sequence in the examined process is sequence `C,V,A,A,S`. Second most frequented is `C,V,A,A,A,A,S`, etc. We can also discover the least frequented sequences that can be identified as some deviations in the process.

Right side of the Fig. 2 shows histogram which is focused to the case type B. There we can see that time (duration of events) influences the amount of different sequences.

Fig. 3 depictures on the left side histogram of case type C and on the right side case type D the sequences with and without time, but with the identification of users (performers of the events). This information also influences the difference between the sequences.

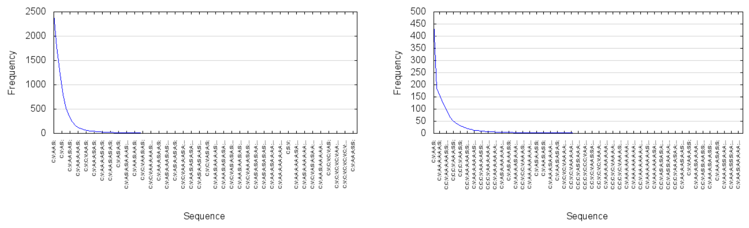

Fig. 2. Left side - histogram Type A, right side - histogram Type B

4.4 Application of sequence alignment methods

We have used four methods to determine the similarity of sequences. Similarity was determined by the equation 1. The main purpose for determining the sequence similarity is that we would like to find similar types of sequences (set

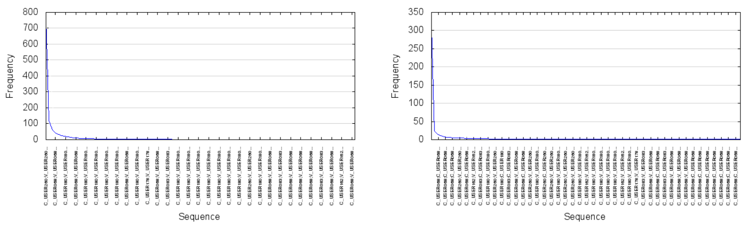

Fig. 3. Left side - histogram Type C, right side - histogram Type D

business processes) as well as the deviations. The four different types of sequences allow us a whole new insight into the performed processes that cannot be done by conventional approaches.

Similarity between sequences was determined by several selected methods. The aim was to find the score, which can show us how similar the event sequences in the case are. The weight distribution in Fig. 4 and Fig. 5 shows us that each method has its specific behavior and due to this the score which determines the similarity between the sequences is different for each method. The advantages and disadvantages describes the following text.

Levenstein distance method does not respect the order of events within the sequences. It only represents amount of necessary steps to change one sequence to another. LCS method respects the events order within the sequences. However, it is suitable, when we can find identical sequences. If the sequences have even a small difference, the similarity weight changes significantly. LCSS method is more tolerant to slight distortions inside the sequences. Of course, it respects the order of events within the sequences. In comparison with LCS, if we add one more event into the sequence, we can find the change of weight similarity, but not as significant then using LCS method. T-WLCS method has as similar behavior as LCSS method. Besides the event order in sequences, T-WLCS method emphasizes the event recurrence within the sequence. However, we must remind that the sequence weight is not similarity for T-WLCS method, it is only a score.

Histograms at Fig. 4 and Fig. 5 show us a distribution of weights for all four case types. These histograms show in fact the applicability of used methods to our example. We can see that the Levensthein and LCS methods are not very suitable for our purposes, because the distribution for these methods does not show much variability, weight distribution is closely concentrated. LCSS and T-WLCS methods seemed to be more promising and we have analyzed the similarities between the sequences in more detail.

Analysis of the result showed that T-WLCS is seems to be more appropriate for our purpose. We have based our decision also on our previous researches that showed suitability of these methods [29, 30].

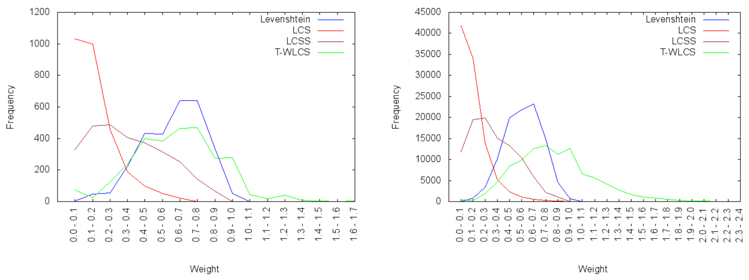

Fig. 4. Histogram - comparison of the methods - Type A(left side), Type B(right side)

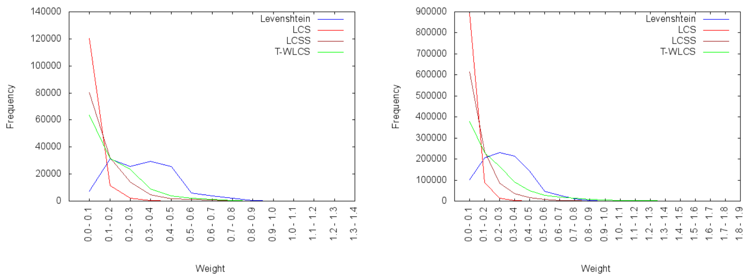

Fig. 5. Histogram - comparison of the methods - Type C(left side), Type D(right side)

4.5 Results

Left side of Fig. 6 visualizes similarity between particular sequences of case type A. Similarity was analyzed by T-WLCS method which we choose like the best representative of the tested methods according to our research that was mentioned in the section 4.4 The structure of the graph show us that where we are able to discover which sequences are more similar than others and what are the connections between them. In the future work we will analyze these parts and we will try to find why is it happening and what it means in the real business.

Right side of Fig. 6 visualizes similarity between particular sequences of case type B. That means there is involved time parameter. We can see that there are some differences according graph on the left side of the Fig. 6. In any case the result is that time parameter have some impact to the examined structure and can show us another dependencies and clusters. That is the interesting result which is in one hand essential and expectable but in the other hand bring us to another view to the examined process that we are not able to see at the beginning of analysis. There are a lot of possibilities to future work that we can do.

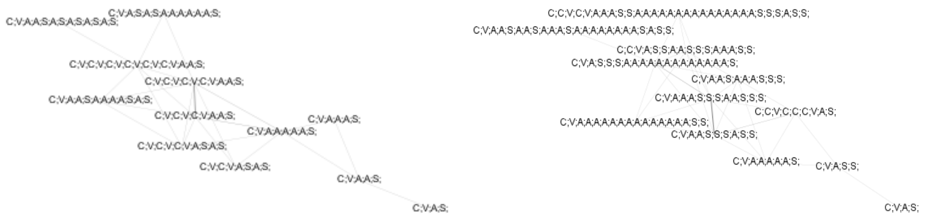
C;V;A;S;A;S;A;A;A;A;A;A;S;
C;V;A;A;S;A;S;A;S;A;S;A;S;A;S;
C;V;C;V;C;V;C;V;C;V;C;V;A;A;S;
C;V;C;V;C;V;C;V;A;A;S;
C;V;A;A;S;A;A;A;A;S;A;S;
C;V;C;V;C;V;A;A;S;
C;V;A;A;A;S;
C;V;A;A;A;A;A;S;
C;V;C;V;C;V;A;S;A;S;
C;V;C;V;A;S;A;S;
C;V;A;A;S;
C;V;A;S;
C;C;V;C;V;A;A;A;S;S;A;A;A;A;A;A;A;A;A;A;A;A;A;A;S;S;S;A;S;S;
C;V;A;A;S;A;A;S;A;A;A;S;A;A;A;A;A;A;A;A;A;S;A;S;S;
C;C;V;A;S;S;A;A;S;S;S;A;A;A;S;S;
C;V;A;S;S;S;A;A;A;A;A;A;A;A;A;A;A;A;A;A;S;
C;V;A;A;S;A;A;A;S;S;S;
C;V;A;A;A;S;S;S;A;A;S;S;S;
C;C;V;C;C;C;V;A;S;
C;V;A;A;A;A;A;A;A;A;A;A;A;A;A;A;A;A;S;S;
C;V;A;A;S;S;S;A;S;S;
C;V;A;A;A;A;A;S;
C;V;A;S;S;
C;V;A;S;

Fig. 6. Left figure shows graph similarity of sequences of sequence distribution Type A (T-WLCS method), Right figure shows Graph similarity of sequences of sequence distribution Type B (T-WLCS method)

5 Conclusion and Future Work

Obtained results show that the proposed approach can be successfully used for the data mining. We can relatively easily reach many types of findings that have to be analyzed then by the customer and the reasons why something is made differently or some process is made in two different ways has to be found.

Our approach allowed us to involve time and user metadata to the examination and we were able to found many interesting results. For example the concrete person behavior can be showed and analyzed what are her/his process instances and the behavior pattern. The goal of our paper was to find out whether we can use the proposed approach to this application domain and how. We have found that the application is possible after the described adjustment and brings us an opportunity to obtain interesting results from the data logs that could not have been seen before by other methods or their execution is difficult.

We would like to continue with the extension of this approach in the future. We want to find out what types of results we can obtain by the usage of different methods, examine the methods and accustom them for the usage on different real examples. Detailed interpretation of different case studies will help us then to determine which method and what views will be used then for data mining and real process examination. The idea is to have a very good control of the process by the usage of the data about already performed process instances.

Acknowledgments.

This work was supported by the project (CZ.1.05/1.1.00/02.0070), funded by the European Regional Development Fund and the national budget of the Czech Republic via the Research and Development for Innovations Operational Programme, by the project New creative teams in priorities of scientific research, reg. no. CZ.1.07/2.3.00/30.0055, supported by Operational Programme Education for Competitiveness and co-financed by the European Social Fund and the state budget of the Czech Republic and by the internal grant agency of VŠB – Technical University of Ostrava, Czech Republic, under the projects no. SP2014/157

"Knowledge modeling, simulation and design of processes" and no. SP2014/154 "Complex network analysis and prediction of network object behavior".

References

1. W.M.P. Van der Aalst, A.J.M.M. Weijters, L. Maruster: Workflow Mining: Discovering Process Models from Event Logs. *Transaction on Knowledge and Data Engineering 16(9)*, 11281142, 2004.
2. W.M.P. Van der Aalst, A.J.M.M. Weijters, Workflow Mining: Process mining: A research agenda *Computers in Industry*, 231-244, 2004.
3. Van Dongen, B.F., De Medeiros, A.K.A., Verbeek, H.M.W., Weijters, A.J.M.M., Van Der Aalst, W.M.P. 2005, "The ProM framework: A new era in process mining tool support", Lecture Notes in Computer Science, pp. 444.
4. E. Esgin, P. Karagoz. Sequence alignment adaptation for process diagnostics and delta analysis, *8th International Conference HAIS*, 191-201, 2013.
5. A. Guo and H. Siegelmann. Time-Warped Longest Common Subsequence Algorithm for Music Retrieval. *In 5th International Conference on Music Information Retrieval ISMIR*, 258261, 2004.
6. D. Gusfield, Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology. *Cambridge University Press*, 2008.
7. D. S. Hirschberg. Algorithms for the Longest Common Subsequence Problem. *J., ACM*, 24:664675, October 1977.
8. Jochen De Weerdt, Annelies Schupp, An Vanderloock, Bart Baesens, Process Mining for the multi-faceted analysis of business processesA case study in a financial services organization, Computers in Industry, Volume 64, Issue 1, January 2013, Pages 57-67, 2012
9. T. Kocyan, J. Martinovič, P. Dráždilová, K. Slaninová, Searching Time Series Based On Pattern Extraction Using Dynamic Time Warping. *In Dateso, Pisek, Czech Republic*, 129-138, 2013.
10. M. Muller. Information Retrieval for Music and Motion. *Springer*, 2007.
11. X. Shi and C. C. Yang. Mining Related Queries from Search Engine Query Logs. *In 15th International Conference on World Wide Web*, 943944, NY, USA, 2006.
12. K. Slaninová, T. Kocyan, J. Martinovič, P. Dráždilová, V. Snášel. Dynamic Time Warping in Analysis of Student Behavioral Patterns. *In Dateso 2012, Zernov, Czech Republic*, Vol. 837, 49-59, 2012.
13. K. Slaninová, J. Martinovič, T. Novosad, P. Dráždilová, L. Vojáček, V. Snášel. Web Site Community Analysis Based on Suffix Tree and Clustering Algorithm. *In IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology - Workshops, WI-IAT, IEEE Computer Society*, 110-113, 2011.
14. K. Slaninová, J. Martinovič, R. Šperka, P. Dráždilová. Extraction of Agent Groups with Similar Behaviour Based on Agent Profiles. *In 12th IFIP TC8 International Conference on Computer Information Systems and Industrial Management, CISIM*, Springer-Verlag, Vol. 8104, 348-357, 2013.
15. J. Štolfa, M. Kopka, S. Štolfa, O. Koberský, V. Snášel. An Application of Process Mining to Invoice Verification Process in SAP, In 4th International Conference on Innovations in Bio-Inspired Computing and Applications, IBICA 2013, 61-74, 2014.

QuickDB – Yet Another Database Management System?[⋆]

Radim Bača, Peter Chovanec, Michal Krátký, and Petr Lukáš

Department of Computer Science, FEECS, VŠB – Technical University of Ostrava
17. listopadu 15, Ostrava, 708 33, Czech Republic
{radim.baca,peter.chovanec,michal.kratky,petr.lukas}@vsb.cz

Abstract. Although DBMS (Database Management Systems) are often hidden for a user, they are a part of many applications utilized in day-by-day life. In general, we can suppose two main types of DBMS: OLTP (On-Line Transaction Processing) and OLAP (On-Line Analytical Processing). We can also distinguish another classification related to the connection of a client and DBMS: client-server and embedded DBMS. The embedded DBMS enable to achieve the maximum performance since an often slow network connection is not used. As a result, they are utilized by in-memory computations where the maximum throughput is required. Although it seems that a lot of high quality DBMS exist and another DBMS are not required, there is not a system to meet all demands of the real world. In this paper, we introduce our prototype of embedded database system called QuickDB. We show that it includes a wide variety of data structures and it provides more efficient performance in many cases compared to up-to-date (embedded) DBMS.

1 Introduction

Although DBMS (Database Management Systems) [9, 6] are often hidden for a user, they are a part of many applications used in day-by-day life. In general, we can suppose two main types of DBMS different in the workload for which they are designed: OLTP (On-Line Transaction Processing) and OLAP (On-Line Analytical Processing). Whereas the first type is mainly used by information systems where a user requires a support of transaction processing [9], the second type is used by, for example, business intelligence applications [15] or some computations and analysis where data structures of DBMS are utilized to manage data. We can also distinguish another classification related to the connection of a client and DBMS: client-server and embedded DBMS. The embedded DBMS enable to achieve the maximum performance since an often slow network connection is not used, therefore, they can be utilized in the case of in-memory computations where the maximum data throughput is required.

The major representatives of the client-server DBMS are the following systems: Oracle Database [18], Microsoft SQL Server [14], MySQL [16], PostgreSQL [21],

[⋆] This work is partially supported by SGS, VŠB – Technical University of Ostrava, No. SP2014/211, Czech Republic.

Firebird [8] and others. The major representatives of the embedded DBMS are the following systems: Microsoft Access[1], SQLite [20], Berkeley DB [17], eXtremeDB[2] and so on. When we consider only relational DBMS, many database systems have appeared during 35 years of their development (see the genealogy of relational DBMS [12]).

Although it seems that a lot of high quality DBMS exist and another DBMS are not required, there is not a system to meet all demands of the real world. For example, Berkeley DB provides the high performance of the B-tree and the paged queue, but it does not support any multidimensional data structure. On the other hand, in the case of libraries including an R-tree implementation (see Table 1), they support especially in-memory implementations, which penalizes them in the case of huge data. Moreover, they do not often support any other data structures. In this paper, we introduce a long-time project of the Database Research Group[3], a prototype DBMS called QuickDB [5]. It includes a wide variety of data structures and it provides more efficient performance in many cases compared to up-to-date DBMS. In Table 1, we show a comparison of individual features of selected DBMS and libraries. We can see that there are only two DBMS supporting all features: QuickDB and SQLite. However, SQLite do not use any cache buffer for data; only the operating system's cache is used during file operations. We must note that the table does not consider all features of DBMS (a support of transaction processing, availability for more platforms, e.g. UNIX and Windows, and so on). Since a performance comparison of embedded and client-server DBMS is rather problematic, QuickDB is compared only with embedded DBMS and libraries.

Table 1. Summary of supported features for all database systems and libraries compared in this article

DBS/Library	B-tree	Paged Queue (Heap table)	R-tree	32 bit	64 bit	In Memory/Disk Only/Cache Buffer
QuickDB [5]	✓	✓	✓	✓	✓	Cache Buffer
Berkley DB [17]	✓	✓	×	✓	✓	Cache Buffer
SQLite [20]	✓	✓	✓	✓	✓	Disk Only
Boost [4]	×	✓	✓	✓	✓	In Memory
libSpatialIndex [11]	×	×	✓	✓	✓	In Memory/ Disk Only
RTreeStar [19]	×	×	✓	✓	✓	In Memory
Superliminal Rtree [7]	×	×	✓	✓	✓	In Memory

[1] `http://office.microsoft.com/en-us/access/`

[2] `http://www.mcobject.com/extremedbfamily.shtml`

[3] `http://db.cs.vsb.cz/`

This paper is organized as follows. In Section 2, we describe an architecture of QuickDB. In Section 3, we put forward a comparison of QuickDB with other DBMS. In the last section, the paper content is resumed and the possibility of a future work is outlined.

2 QuickDB

In Figure 2, we see an architecture of QuickDB[4]. The cache buffer preserves pages of data structures to prevent disk accesses when a page is required. The overhead of the page size compared to the size on a disk is approximately 20%, i.e. the in-memory page size is 9,830B in the case of the 8kB page size. When a data structure operation returns a result, e.g. the range query, the result is stored in a ResultSet and returned to a user. After the user closes the ResultSet, it is returned to QuickDB.

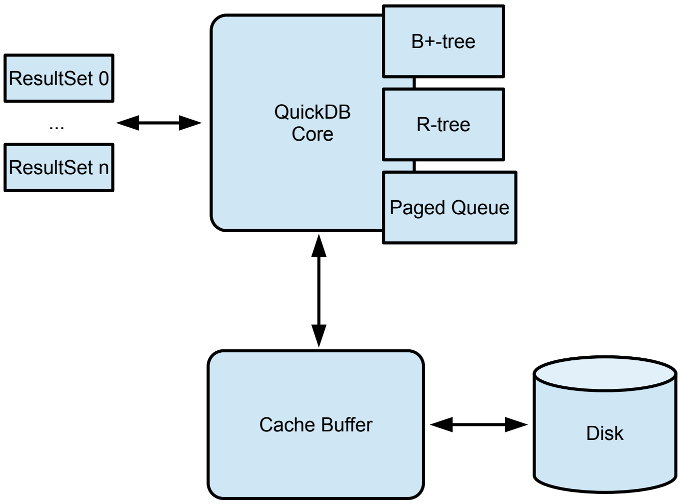

Fig. 1. An architecture of QuickDB

There exists an implementation of the B-tree, R-tree, and paged queue utilizing the core of QuickDB. The goal of QuickDB is to provide the maximum performance instead of an implementation of rich functionalities like a support of methods stored in a database and so on.

[4] QuickDB is implemented in C++.

3 Comparison of DBMS

3.1 Testing Environment

We perform a set of various tests where we compare our QuickDB with state-of-the-art implementations in each area. By the term area we mean a data structure type. Moreover, in Section 3.4, we show results of an application of QuickDB – a native XML database.

All experiments are executed in a single thread on an Intel Xeon 2.9 GHz CPU. We use a real data set of meteorological measurements of the Czech Hydrometeorological Institute[5], where the dimension of each record is 5. The collection contains 57,852,305 records and the total size of its textual representation is 1.12 GB. Queries used during the tests are randomly generated.

3.2 Comparison of Basic Data Structures

In this test, we compare the performance of two QuickDB basic data structures: the B-tree and the heap table (paged queue). The B-tree is a traditional data structure used by all relational DBMS [2]. The heap table is also supported by all relational DBMS and we test just a sequential scan in the heap table using a cursor.

B-tree We compare the QuickDB's B-tree performance of the insert and point query operations with the Berkeley DB and SQLite here. The performance of the insert operation can be influenced by several different aspects where we consider the following: (1) whether the data fit into the main memory cache or not, and (2) whether the data are sorted before the insert or not. All combinations for each B-tree are tested and the results are summarized in Table 2.

We can observe that SQLite performs worst in all cases. An overhead of SQLite is probably induced by the SQL interface and lack of its own buffer cache. Clearly, QuickDB and Berkeley DB have very similar performance on our data set, however, QuickDB slightly outperforms Berkeley DB in all tests. The only disadvantage of QuickDB is its a slightly bigger index.

Heap Table (Paged Queue) Now we compare the performance of the QuickDB's heap table with the Berkeley DB's queue. We use the Berkeley DB's queue since it supports the insert and sequential scan operations. The insert operation is preformed with limited memory, however, it is not very important here since only a sequential write is utilized. Then we perform the sequential scan with both cold and warm caches. The term warm cache means that all the data are already stored in the main memory.

Results are shown in Table 3. QuickDB is twice faster during the insert than Berkeley DB. Surprisingly, the performance of the Berkeley DB queue scan is

[5] `http://www.chmi.cz/`

Table 2. The B-tree comparison

	Operations	**QuickDB**	**Berkeley DB**	**SQLite**
80% of data fit in the main memory	Sorted insertion [thousands per second]	**531**	445	173
	Random insertion [thousands per second]	**289**	264	21
All data fit in the main memory	Sorted insertion [thousands per second]	**674**	526	173
	Random insertion [thousands per second]	**357**	312	21
	Queries [s]	**0.35**	0.38	0.64
	Index size [GB]	3.5	3.39	**2.57**

the same no matter whether the cache is cold or not. It leads us to a conclusion that Berkeley DB uses the OS file system cache which influences the results. However, QuickDB significantly outperforms Berkeley DB during in-memory sequential scan. On the other hand, the size of the Berkeley DB index file is quarter of the data set size, whereas, the QuickDB index file is equal to the data size.

Table 3. The heap table (page queue) comparison

	QuickDB	**Berkeley DB**
Inserting [thousands per second]	**2,244**	1,468
Sequential scan (warm) [s]	**4.64**	21.1
Sequential scan (cold) [s]	27.11	**21.15**
Index size [GB]	1.12	**0.25**

3.3 Comparison of Multidimensional Data Structures

In this section, we compare a performance a multidimensional data structure called the R-tree [10, 3] implemented in QuickDB with some other existing implementations. We compare the performance of both insert and range query operations. In the experiments, we use three collections of queries (a detail description is shown in Table 4).

The performance of inserting and query processing is compared with several libraries, namely Boost [4], libSpatialIndex [11], RTreeStar [19], Superliminal Rtree [7], and SQLite [20] as a representative of embedded database systems. Relational DBMS based on a client-server architecture (Oracle, PostgreSQL, MySQL, and so on) have not been taken into account since it is rather problematic to compare embedded and client-server DBMS.

Table 4. A basic characteristic of query groups

Query Group	**Result Size**	**Avg. Result Size**
1	1	1.0
2	⟨2, 999⟩	302.5
3	⟨10000, 99,999⟩	45,172

Since all tested libraries (except QuickDB and libSpatialIndex) support only an in-memory storage, no secondary storage has been used during the experiments and we do not measure the size of the indices. The page size is 8kB in all cases. In Table 5, we see that QuickDB and Boost provide the highest performance. While Boost is more efficient in the case of the insert operation, QuickDB slightly outperforms it in the case of query processing. We see that other implementations are outperformed by QuickDB and Boost.

Table 5. The R-tree comparison

DBMS/Library	**Insert Time** [thousands of inserts/s]	**Query Processing Time** [thousands of quries/s]		
		QG1	**QG2**	**QG3**
QuickDB	68.3	**38.9**	19.8	**1.7**
Boost	**97.2**	33.3	**20.0**	1.40
libSpatialIndex	12.2	18.5	13.2	0.74
RTreeStar	56.53	20.8	14.7	0.43
Superliminal Rtree	70.2	10.0	6.60	1.30
SQLite	66.4	17.30	12.10	0.76

3.4 Comparison of QuickDB Application

In this section, we present an application of the QuickDB framework. It is a native XML database called QuickXDB introduced in [13]. We briefly discuss how the QuickDB framework is exploited here and compare it with some other solutions.

Data structures There are two indexes representing an XML data collection, namely *document index* and *partition index* [1]. The document index serves as a primary access path since it fully describes both tree structure and actual text values of an XML collection[6]. We exploit two persistent data structures of the QuickDB framework: the B-tree and the paged queue.

[6] The reader is expected to understand the terms of the XML tree data model.

The B-tree of the document index has a *node label* as a key. There are two fundamental purposes of the node label: (1) it uniquely identifies each XML data node and (2) we are able to resolve a structural relationship of two nodes, e.g., one is a parent of another. The leaf nodes of the B-tree contain tags (node names) of XML nodes and pointers into the paged queue where text values are stored. The records for XML nodes in the leaf nodes of the B-tree are stored in the same order as in the original XML document or collection. Consequently, we can profit from using range queries to obtain the whole subtree of an XML node.

The partition index is the secondary access path. It is also a combination of the B-tree and the paged queue. However, here a tag name serves as a key and items of the B-tree leaf nodes include pointers to paged queues where corresponding node labels are stored in the document order.

Experimental evaluation We have performed a time comparison with 2 other native XML databases: MonetDB[7] and BaseX[8]. We have picked up 4 data collections: XMark[9] with factor $f = 10$ (1.1 GB), Swissprot (109 MB), TreeBank (82 MB), and DBLP (127 MB)[10].

For each collection, we have generated 3 sets of 50 distinct random XPath queries oriented purely on searching structural relationships between XML nodes. The sets differ in the upper range of their *selectivity*[11]: (1) selectivity up to 100%, (2) selectivity up to 10%, and (3) selectivity up to 1%. Different query selectivity can cause a different utilization of indexes. The complexity of queries vary for a different number of location steps (from 2 to 11) and also for a different number of branching predicates (from 0 to 4). Both ancestor-descendant and parent-child tree axes are randomly used. Branching predicates contain either single XPath subqueries or more subqueries connected by a logical conjunction.

Summarized results are presented in Figure 2. The values on the vertical axis stand for the total processing time of a set of queries. Each single query was evaluated 5 times, only arithmetic means of 3 times (without the best and the worst case) are considered.

We can see that our QuickXDB outperforms BaseX on all query sets except XMark with selectivity up to 1% and 10%. We also evaluate queries with the higher selectivity (up to 1% and 10%) on DBLP, Swissprot, and TreeBank collections faster than MonetDB. On the Swissprot query set with selectivity up to 1%, QuickXDB is more than 20× faster than BaseX and 8× faster than MonetDB. On the other hand, our QuickXDB is approximately 2.5× slower than MonetDB on DBLP and XMark query sets with selectivity up to 100% and 3.3× slower than BaseX on XMark with selectivity up to 10%.

[7] http://www.monetdb.org/XQuery/
[8] http://www.basex.org
[9] http://www.xml-benchmark.org/
[10] http://www.cs.washington.edu/research/xmldatasets/www/repository.html
[11] If we for example have a query `//a[./b]//c`, selectivity can be computed as `count(//a[./b]//c) div count(//c)`

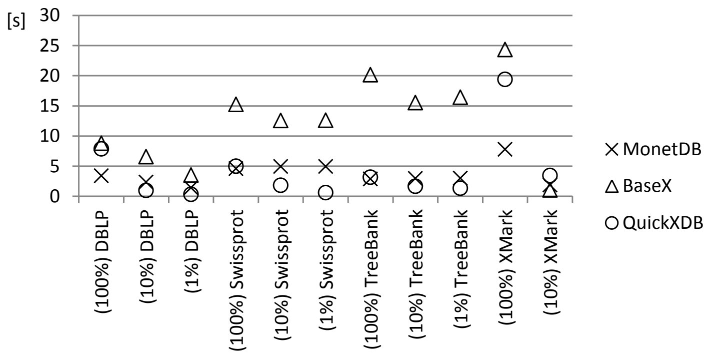

Fig. 2. A comparison of native XML databases

4 Conclusion

In this paper, we introduced a comparison of our prototype of embedded database system called QuickDB with other up-to-date embedded DBMS and libraries. As mentioned, the main goal of QuickDB is to provide the maximum performance. The results show that QuickDB outperforms these DBMS and libraries in a lot of cases. In our future work, we want to compare other features of DBMS, for example, scalability performance, transaction processing performance and so on.

References

1. R. Bača and M. Krátký. XML Query Processing Efficiency and Optimality. In *Proceeding IDEAS 12 Proceedings of the 16th International Database Engineering & Applications Symposium*, pages 8–13. ACM, 2012.
2. R. Bayer and E. M. McCreight. Organization and Maintenance of Large Ordered Indices. *Acta Inf.*, 1:173–189, 1972.
3. N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD 1990)*, 1990.
4. Boost.org. Boost C++ Libraries, `http://www.boost.org/`, 2014.
5. Database Reasearch Group. QuickDB, `http://db.cs.vsb.cz/`, 2014.
6. C. Date. *An Introduction to Database Systems.* Addison-Wesley, 8th edition, 2003.
7. G. Douglas. Superliminal Rtree, `http://superliminal.com/sources/sources.htm#C%20&%20C++%20Code`, 2014.
8. Firebird Foundation Incorporated. Firebird, `http://www.firebirdsql.org/`, 2014.
9. H. Garcia-Molina, J. Ullman, and J. Widom. *Database Systems: The Complete Book.* Prentice Hall, 2nd edition, 2008.
10. A. Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD 1984)*, pages 47–57. ACM Press, June 1984.

11. M. Hadjieleftheriou. libSpatialIndex, `http://libspatialindex.github.io/`, 2013.
12. Hasso-Plattner-Institut. The HPI Genealogy of Relational Database Management Systems, `http://www.hpi.uni-potsdam.de/naumann/projekte/rdbms_genealogy.html`, 2014.
13. P. Lukáš, R. Bača, and M. Krátký. QuickXDB: A Prototype of a Native XML DBMS. In *Proceedings of the Dateso 2013 Annual International Workshop*, pages 36–47, 2013.
14. Microsoft. Microsoft SQL Server 2012, `http://www.microsoft.com/en-us/sqlserver/default.aspx`, 2014.
15. S. Negash. Business Intelligence. *Communications of the Association for Information Systems*, 13, `http://aisel.aisnet.org/cais/vol13/iss1/15`, 2004.
16. Oracle. MySQL Community Edition, `http://www.mysql.com/products/community/`, 2014.
17. Oracle. Oracle Berkeley DB 12c, `http://www.oracle.com/technetwork/database/berkeleydb`, 2014.
18. Oracle. Oracle Database 12c, `http://www.oracle.com/us/products/database/overview/index.html`, 2014.
19. D. Spicuzza. R* Tree Implementation for C++, `http://www.virtualroadside.com/blog/index.php/2008/10/04/r-tree-implementation-for-cpp/`, 2014.
20. SQLite Consortium. Sqlite, `http://www.sqlite.org/`, 2014.
21. The PostgreSQL Global Development Group. PostgreSQL, `http://www.postgresql.org/`, 2014.

Author Index