RASLAN 2014
Recent Advances in Slavonic Natural Language Processing

librix.eu

A. Horák, P. Rychlý (Eds.)

RASLAN 2014

Recent Advances in Slavonic Natural Language Processing

**Eighth Workshop on Recent Advances
in Slavonic Natural Language Processing,
RASLAN 2014
Karlova Studánka, Czech Republic,
December 5–7, 2014
Proceedings**

Tribun EU
2014

Proceedings Editors

Aleš Horák
Faculty of Informatics, Masaryk University
Department of Information Technologies
Botanická 68a
CZ-602 00 Brno, Czech Republic
Email: hales@fi.muni.cz

Pavel Rychlý
Faculty of Informatics, Masaryk University
Department of Information Technologies
Botanická 68a
CZ-602 00 Brno, Czech Republic
Email: pary@fi.muni.cz

ISSN 2336-4289

Preface

This volume contains the Proceedings of the Eighth Workshop on Recent Advances in Slavonic Natural Language Processing (RASLAN 2014) held on December 5–7, 2014 in Karlova Studánka, Czech Republic.

The RASLAN Workshop is an event dedicated to the exchange of information between researchers working on the projects of computer processing of Slavonic languages and related areas going on in the NLP Centre at the Faculty of Informatics, Masaryk University, Brno. RASLAN is focused on theoretical as well as technical aspects of the project work, on presentations of the verified methods together with descriptions of development trends. The workshop also serves as a place for discussions about new ideas. The intention is to have it as a forum for presentation and discussion of the latest developments in the field of language engineering, especially for undergraduates and postgraduates affiliated to the NLP Centre at FI MU.

Topics of the Workshop cover a wide range of subfields from the area of artificial intelligence and natural language processing including (but not limited to):

- text corpora and tagging,
- syntactic analysis,
- sense disambiguation,
- machine translation, computer lexicography,
- semantic networks and ontologies,
- semantic web,
- knowledge representation,
- logical analysis of natural language,
- applied systems and software for NLP.

RASLAN 2014 offers a rich program of presentations, short talks, technical papers and mainly discussions. A total of 18 papers were accepted, contributed altogether by 24 authors. Our thanks go to the Program Committee members and we would also like to express our appreciation to all the members of the Organizing Committee for their tireless efforts in organizing the Workshop and ensuring its smooth running. In particular, we would like to mention the work of Aleš Horák, Pavel Rychlý and Lucia Kocincová. The TEXpertise of Adam Rambousek (based on LATEX macros prepared by Petr Sojka) resulted in the extremely speedy and efficient production of the volume which you are now holding in your hands. Last but not least, the cooperation of Tribun EU as both publisher and printer of these proceedings is gratefully acknowledged.

Brno, December 2014 Karel Pala

Table of Contents

I Language Modelling

II Text Corpora

III Semantics and Information Retrieval

IV Morphology and Lexicon

Part I

Language Modelling

Character-based Language Model

Vít Baisa

Natural Language Processing Centre
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
xbaisa@fi.muni.cz

Abstract. Language modelling and also other natural language processing tasks are usually based on words. I present here a more general yet simpler approach to language modelling using much smaller units of text data: character-based language model (CBLM).[1] In this paper I describe the underlying data structure of the model, evaluate the model using standard measures (entropy, perplexity). As a proof-of-concept and an extrinsic evaluation I present also a random sentence generator based on this model.

Keywords: language model, suffix array, LCP, trie, character-based, random text generator, corpus

1 Introduction

Current approaches to language modelling are based almost utterly on words. To work with words, the input data needs to be tokenized which might be quite tricky for some languages. The tokenization might cause errors which are propagated to following processing steps. But even if the tokenization was 100% reliable, another problem emerges: word-based language models treat similar words as completely unrelated. Consider two words *platypus* and *platypuses*. The former is contained in the latter yet they will be treated completely independently. This issue can be sorted out partially by using factored language models [1] where lemmas and morphological information (here singular vs. plural number of the same lemma) are treated simultaneously with the word forms.

In most systems, word-based language models are based on n-grams (usually 3–4) and on Markov chain of the corresponding order where only a finite and fixed number of previous words is taken into account. I propose a model which tackles with the above-mentioned problems. The tokenization is removed from the process of building the model since the model uses sequences of characters (or bytes) from the input data. Words (byte sequences) which share prefix of characters (bytes) are stored on the same place in the

[1] I call this ChaRactEr-BasEd LangUage Model (CBLM) *cerebellum*: a part of human brain which plays an important role in motor control and which is involved also in some cognitive processes including language processing.

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2014, pp. 3–10, 2014. © Tribun EU 2014

model. The model uses suffix array and trie structure and is completely language independent.

The aim of CBLM is to make language modelling more robust and at the same time simpler: with no need for interpolating, smoothing and other language modelling techniques.

2 Related work

Character-based language models are used very rarely despite they are described frequently in theoretical literature. That is because a standard n-gram character-based language models would suffer from very limited context: even 10-grams are not expressive enough since they describe only very limited width of context. Even the famous Shannon's paper [2] mentions a simple uni-, bi- and tri-gram models but then it swiftly moves to word-based models.

There have been some attempts to use sub-word units (morphemes) for language modelling, especially for speech recognition tasks [3] for morphologically rich languages like Finnish and Hungarian but they have not gone deeper.

Variable-length n-gram modelling is also closely related but the model described in [4] is based rather on categories than on substrings from the raw input data. Suffix array language model (SALM) based on words has been proposed in [5].

3 Building the model

As input, any plain text (in any encoding but it is most convenient to use UTF-8) can be used. In a previous version of the model all input characters were encoded to a 7-bit code (the last bit was used for storing structure information). Currently the model requires a simpler preprocessing: the data is taken as is—as a sequence of bytes. Sentences are separated by a newline character. The only pre-processing is lower-casing—quite common practice.

3.1 Suffix array, longest common prefix array

The next step is suffix array construction. Suffix array (SA) is a list of indexes (positions) in the input data which are sorted according to lexicographical order of suffixes starting at the corresponding positions in the data. The Table 1 shows an example of a SA constructed for string *popocatepetl*. The resulting SA is in the third column. The last column contains longest common prefix array (LCP) which corresponds to a number of common characters between two consecutive suffixes in the SA.

I use *libdivsufsort*[2] library for fast SA construction in $O(n \log n)$ time where n is input data size in bytes. The size of an input is limited to 2 GB since longer

[2] `https://code.google.com/p/libdivsufsort/`

Table 1: Suffix array example

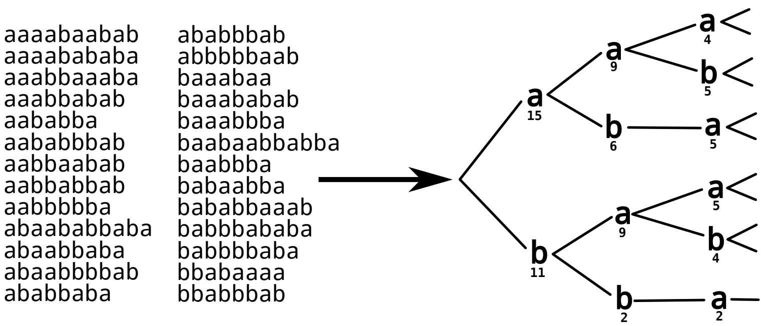
I	suffix	SA	sorted suffix	LCP
0	popocatepetl	5	atepetl	0
1	opocatepetl	4	catepetl	0
2	pocatepetl	7	epetl	0
3	ocatepetl	9	etl	1
4	catepetl	11	l	0
5	atepetl	3	ocatepetl	0
6	tepetl	1	opocatepetl	1
7	epetl	8	petl	0
8	petl	2	pocatepetl	1
9	etl	0	popocatepetl	2
10	tl	6	tepetl	0
11	l	10	tl	1

data could not be encoded using 4-byte integer indexes. The size of a compiled SA is $O(n \log n)$.

LCP is computed separately using an inverse SA in $O(n)$ time and $O(n)$ space. To limit the size of LCP array, the highest possible number in LCP array is 256 (1 B per item). Thus the longest substring which can be stored in the model has length 256 characters (bytes).

3.2 Trie

Once SA and LCP are built, all prefixes from SA which occur more than $N\times$ are put into trie structure. The N is the only parameter used in construction of the trie. Each node in the trie stores probability (relative frequency) of occurrence of the corresponding prefix in SA.



aaaabaabab
aaaabababa
aaabbaaaba
aaabbabab
aabababba
aababbbab
aabbaabab
aabbabbab
aabbbbba
abaababbaba
abaabbaba
abaabbbbab
ababbaba
ababbbab
abbbbbaab
baaabaa
baaababab
baaabbba
baabaabbabba
baabbba
babaabba
bababbaaab
babbbababa
babbbbaba
bbabaaaa
bbabbbab

Fig. 1: Construction of a trie from an example suffix array

In Figure 1 you can see an example of suffix array turned into trie structure. Only the upper part of the trie is shown. The numbers below the nodes correspond to frequencies of the prefixes.

The trie is stored in a linear list—the tree structure of the trie is preserved using integer indexes of the list. Each item of the list stores 4 slots: 1) an address of the eldest children, 2) probability of the current prefix, 3) the character byte itself and 4) binary true or false: if the current item is the last node in a row of siblings.

The siblings are sorted according probability. In Figure 2 there is an example for substring *barb* from a Czech model. It is obvious that after the prefix, characters *a, o, i* and *e* are the most frequent. They occur in words like *barbar, barbora, barbie, barbecue, rebarbora* etc. The dots in the Figure mean a space skipped between the trie items (nodes). After substring *barbo*, the most probable characters are *r* (75%) and in *ř* (22%). See the last two items in Figure 2. Character *ř* is in fact represented by two nodes: a node with byte value 197 and its children node (153) but here I have simplified it.

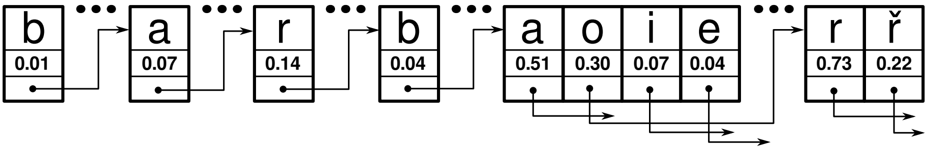


Fig. 2: Trie structure

4 Language model

A language model is a prescription for assigning probabilities to an input data (in this case a sequence of bytes). Here I present a straightforward algorithm using the trie as an underlying data structure. It is important to emphasize that this approach is only a first attempt at language modelling using the trie described above. Further improvements are to be implemented.

Each node in a trie contains probability distribution of all following characters (bytes). The longer is the path from root to a node, the more accurate is the probability distribution (and also the lower entropy and perplexity) in the node since a longer path means a longer context. Following is description of the algorithm: how to compute a probability of any sequence of bytes.

The algorithm starts from the first byte in the data and from the root of the trie. If the byte is among the root's children, the initial probability 1.0 is multiplied by the corresponding probability stored in the relevant child node. The algorithm stores the position of the child node and repeats the procedure for next bytes in the input data. If a current byte is not found among children at a current position, the current prefix (defined by a path from the root to the current position) is shortened from the left (by one byte) and the shortened path is translated to the root (the same character bytes but a different path). It holds that if a path (sequence of bytes) is found wherever in the trie then it must be translatable to the root of the trie. It is a property of the original suffix array. After the translation, the lookup is repeated. If the byte is not found, the path is shortened from the left again and translated again until the byte is found among descendants of the last node in the translated path. It may occur that the path is shortened to an empty path. In that case the procedure continues from the root as at the beginning of the algorithm. Every time a byte from the input data is found among a children of a current position, the overall probability is multiplied by the probability of the children.

It is necessary that probability of any subsequence of the input data is greater than 0 otherwise the result probability would be zero too. In n-gram models the solution is achieved by smoothing language models using techniques designed like Katz, Kneser-Ney or Good-Turing smoothing. In CBLM, the problem of zero probability (caused by symbols which are in the input data but do not occur in an input data more than $N\times$) is solved by assigning probability p_r to all unseen symbols. Probability p_r is taken from the first level in trie—the complement of the sum of probabilities of all child nodes of the root. For one English model trained on Orwell's *1984* $p_r = 0.000018$ since some symbols (+, %) occurred less than $N\times$.

The described procedure above can be slightly modified to obtain a random text generator (see Section 6). The bytes are not read from the input but generated randomly from distribution probabilities stored in nodes.

5 Intrinsic evaluation: entropy per byte

The standard way to evaluate language models is to measure entropy per word. In the case of this model I use entropy H and perplexity PP per byte. The formulas for test data $b_1 \cdots b_N$ and model m are as follow:

$$H(m) = -\frac{1}{N}\sum_{i=1}^{N} \log p(b_i)$$

$$PP = 2^{H(m)}$$

where N is length of the test data and $p(b_i)$ is probability given by the algorithm.

Table 2 shows results for English and Czech data. The models for English has been built from British National Corpus and from George Orwell's *1984*. The test data were Lewis Carroll's *Alice in Wonderland* and George Orwell's *1984*. The models for Czech has been built from the Czech National Corpus, Czech Wikipedia corpus (csWiki) and Czech Web corpus czTenTen[3] [6] (csWeb).

[3] http://www.sketchengine.co.uk/documentation/wiki/Corpora/czTenTen2

Table 2: Evaluation of Czech and English models on Lewis Carroll's *Alice in Wonderland* and George Orwell's *1984*.

Model	Test	Size	H	PP
BNC_2	Alice	330 M	2.286	4.879
BNC_3	Alice	212 M	2.294	4.904
BNC_2	1984	330 M	**1.664**	3.170
BNC_3	1984	212 M	1.671	3.184
$\mathrm{SYN2000}_4$	1984	362 M	1.837	3.574
csWiki_5	1984	300 M	1.850	3.607
csWeb_4	1984	312 M	**1.571**	2.972

Some observations from the tables follow. The BNC_2 model has achieved only a slightly better entropy and perplexity than BNC_3 for both test data. Notable is also the fact that both models assign considerably higher entropy and perplexity to *Alice*. It is probably caused by the peculiar language of Carroll. For comparison—the entropy of English has been estimated to 1.75 [7]. The best Czech model is csWeb_5 which obtained 1.571 entropy for Orwell's *1984*.

The performance of the Czech and English models is quite stable. When Markov model *N* parameter is fixed, performance (perplexity) differs substantially when languages from different language families are evaluated. See also the comparable performance of Hungarian and English random generator in Section 6. Further intrinsic evaluation is to be carried out using a standard statistical language modelling benchmark, e.g. one billion word benchmark [8] or Brown corpus.

6 Extrinsic evaluation: random sentence generator

It has been reported that a language model perplexity measure is not correlating well with the evaluations of applications in which the model is used (e.g. [9]). That is why some researchers prefer extrinsic evaluation methods over intrinsic measures. To provide an extrinsic evaluation of the presented model I have developed a simple random text generator based on CBLM.[4] It offers models for English, Georgian, Hungarian, Japanese, Russian, Turkish, Slovak, Czech and Latin. Users may save generated sentences (*Like* link) which are then available as favourite sentences at the bottom of the web page.

The algorithm is a modification of the language model algorithm. It generates a random stream of bytes and whenever it generates a second new line character, the result is written to the output. By using the sequence of bytes between two newlines, the generator is capable of generating texts roughly on sentence level.

[4] `http://corpora.fi.muni.cz/cblm/generate.cgi`

To increase legibility, initial letters and named entities in the following example sentences have been upper-cased. The source data for the examples were as follow. English: British National Corpus, Czech: czTenTen (first 600 M tokens), Hungarian: Hungarian Wikipedia, Latin: a Latin corpus and Slovak: Slovak Wikipedia. In almost all cases $N = 3$.

English First there is the fact that he was listening to the sound of the shot and killed in the end a precise answer to the control of the common ancestor of the modern city of Katherine street, and when the final result may be the structure of conservative politics; and they were standing in the corner of the room.

Czech Pornoherečka Sharon Stone se nachází v blízkosti lesa. ¶ Máme malý byt, tak jsem tu zase. ¶ Změna je život a tak by nás nevolili. ¶ Petrovi se to začalo projevovat na veřejnosti. ¶ Vojáci byli po zásluze odměněni pohledem na tvorbu mléka. ¶ Graf znázorňuje utrpení Kristovo, jež mělo splňovat následující kritéria.

Hungarian Az egyesület székhelye: 100 m-es uszonyos gyorsúszásban a következő években is részt vettek a díjat az égre nézve szójaszármazékot. ¶ Az oldal az első lépés a tengeri akvarisztikával foglalkozó szakemberek számára is ideális szállás költsége a vevőt terhelik.

Slovak Jeho dizajn je v zrelom veku, a to najmä v prípade nutnosti starala sa o pomoc na rozdiel od mesta prechádza hranica grófstva Cork. ¶ V roku 2001 sa začala viac zameraný na obdobie 2006 prestúpil do monastiera pri rieke Marica, ktorú objavil W. Herschel 10. septembra 1785.

Latin Quinto autem anno andum civitates et per grin in multitudinem quae uerae e aeque ad omne bonum. ¶ Augustinus: video et oratteantur in caelum clamor eus adiutor meus, et uit, quam hoc crimen enim contentit et a debent.

7 Future work & conclusion

I have described a first approximation of the language model. In the future I want to exploit more some properties of the trie model, e.g. probabilities of sequences which follow a given sequence (not necessarily immediately following it). This would allow to express connections (associations) between any two byte sequences and to capture a broader context.

Acknowledgement This work was partially supported by the Ministry of Education of CR within the LINDAT-Clarin project LM2010013 and by the Czech-Norwegian Research Programme within the HaBiT Project 7F14047.

References

1. Bilmes, J.A., Kirchhoff, K.: Factored language models and generalized parallel backoff. In: Proceedings of the 2003 Conference of the North American Chapter

of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003–short papers-Volume 2, Association for Computational Linguistics (2003) 4–6

2. Shannon, C.: A mathematical theory of communication. Bell Sys. Tech. J. **27** 379–423
3. Creutz, M., Hirsimäki, T., Kurimo, M., Puurula, A., Pylkkönen, J., Siivola, V., Varjokallio, M., Arisoy, E., Saraçlar, M., Stolcke, A.: Morph-based speech recognition and modeling of out-of-vocabulary words across languages. ACM Transactions on Speech and Language Processing (TSLP) **5**(1) (2007) 3
4. Niesler, T.R., Woodland, P.: A variable-length category-based n-gram language model. In: Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on. Volume 1., IEEE (1996) 164–167
5. Zhang, Y., Vogel, S.: Suffix array and its applications in empirical natural language processing. Technical report, Technical Report CMU-LTI-06-010, Language Technologies Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA (2006)
6. Suchomel, V.: Recent czech web corpora. In: 6th Workshop on Recent Advances in Slavonic Natural Language Processing. Brno
7. Brown, P.F., Pietra, V.J.D., Mercer, R.L., Pietra, S.A.D., Lai, J.C.: An estimate of an upper bound for the entropy of english. Comput. Linguist. **18**(1) (March 1992) 31–40
8. Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P.: One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling. arXiv preprint arXiv:1312.3005. 2013.
9. Iyer, R., Ostendorf, M. Meteer, M.: Analyzing and predicting language model improvements. Automatic Speech Recognition and Understanding, 1997. 254–261

A System for Predictive Writing

Zuzana Nevěřilová and Barbora Ulipová

Computational Linguistics Centre
Faculty of Arts, Masaryk University
Arne Nováka 1, 602 00 Brno, Czech Republic
xpopelk@fi.muni.cz, b.ulipova@gmail.com

Abstract. Most predictive writing systems are based on n-gram model with different size. Systems designed for English are easier than those for flective languages since even smaller models allow reasonable coverage. However, the same corpus size is significantly insufficient for languages with many word forms. The paper presents a new predictive writing system based on n-grams calculated from a large corpus.

We designed the high-performance server-side script that returns either the most probable endings of a word or the most probable following words. We also designed the client-side script that is suitable for desktop computers without touchscreens.

We calculated 150 millions most frequent n-grams for $n = 1, \ldots, 12$ from a Czech corpus and evaluated the writing system on Czech texts. The system was then extended by custom-built model that can consist of domain or user specific n-grams. We measured the key stroke per character (KSPC) rate in two different modes: one – called *letter KSPC* – excludes the control keys since they are input method specific, the other – called *real KSPC* – includes all key strokes. We have shown that the system performs well in general (letter KSPC on average was 0.64, real KSPC on average was 0.77) but performs even better on specific domains with the appropriate custom-built model (letter KSPC and real KSPC were on average 0.63 and 0.73 respectively).

The system was tested on Czech, however it can easily be adapted an arbitrary language. Due to its performance, the system is suitable for languages with high inflection.

Keywords: predictive writing, n-gram language model, corpus, KSPC

1 Introduction

Predictive writing is a useful and popular feature embedded in modern web browsers and cell phones where either endings of an unfinished word or a following word are suggested to the user. Predictive writing is used in order to save the number of key strokes and prevent spelling errors. Some users may even use it to find the correct spelling of words they are not sure about. For these reasons, predictive writing applications are not only useful for cell phones and tablets but they can also support writing on conventional keyboards.

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2014, pp. 11–18, 2014. © Tribun EU 2014

We present a predictive writing web application that is suitable for devices with conventional keyboards. The application is based on the client-server architecture and thus it can benefit from server perfomance: the language model can be much larger (and thus more precise) than language models that have to use the limited memory space of mobile devices.

This paper is organized as follows: Section 2 overviews in short the related work, in Section 3, we present our system and its extension with custom-built model. In Section 4, we describe the settings of the evaluation experiment and the experiment itself. Section 5 discusses the results. In Section 6, we propose further work for the future.

2 Related Work

Prediction is well established in many tasks e.g. in web browsers and search engines: a web browser usually keeps the search history, and search engines use data obtained from their users as well, e. g. Google uses a combination of phrases from search history of a particular user and overall search history. In addition, it adds information from Google+ profiles[1]. The aim of this prediction is to check spelling and reduction of number of characters the user has to type.

Historically, predictive writing became popular on cell phones with numeric keyboard. Users became accustomed to cluster keyboards (i.e. keyboards with highly ambiguous keys) such as Tegic T9 or Motorola iTap. For example, [1] used vocabularies of 10k, 40k, 160k and 463k words (i.e. unigrams) together with up to 5 milion n-grams (for $n = 2, 3$). The authors concluded that n-gram models are more robust than unigram models (such as T9).

Predictive keyboards on today's mobile devices serve a slightly different purpose: the typing error rate on software keyboards is higher than on hardware keyboards. [2] reported "average number of errors on software keyboards 4.55%, and the average number of hardware was 1.36%". It is thus more comfortable when users do not have to type much.

[3] proposes a simple measure to characterize text entry techniques: key strokes per character (KSPC) "is the number of key strokes required, on average, to generate a character of text for a given text entry technique in a given language". The exact calculation of KSPC depends on both hardware (i.e. it is different on touchscreen keyboards, hardware keyboards, or stylus tapping) and software (i.e. how many characters the user has to tap before the desired word is available to select). For this reason, we calculated two times: KSPC excluding the control keys (the arrows) and KSPC including all keys. In this paper, we call the measures *letter KSPC* and *real KSPC* respectively. The letter KSPC includes the tab key since it is the key stroke that leads to selection of a particular word.

Since the cited text input techniques were primarily developed for English, less attention is paid to non-English texts. For example, [4, p. 9] only state

[1] `https://support.google.com/websearch/answer/106230?hl=en`

that entry of characters not present in the English alphabet increases the number of key strokes required. [4] have shown that n-gram based models for statistical prediction are less reliable for inflected languages but "still offer quite reasonable predictive power".

3 The Predictive Writing System

3.1 Server-side design

We implemented a predictive writing server-side script that benefits from n-grams calculated from the czTenTen corpus. This corpus is currently one of the largest corpora for Czech[2]. The data were collected from different websites, cleaned and deduplicated by the corpora tools [5]. Since the data source contains mainly texts that were not proof-read, the n-grams do not necessarily contain only correct Czech. This issue can be serious when using predictive writing for spelling purposes.

We calculated bigrams using the `lscbgr` tool and filtered out all bigrams with minimum sensitivity < 0.0001. We calculated n-grams for $n = 3, \ldots, 12$ using the `lscngr` tool, then we filtered out all n-grams with frequency of the respective (n-1)-gram representing the n-gram without its last token ($freq_{n-1}$) less than 10. For each such n-gram, we then calculated the score as $n \cdot freq_{n-1}$. in order to prefer longer n-grams. For unigrams, we used the `lsclex` tool[3] and we took all tokens with frequency greater than one and length smaller than 30 characters. All the mentioned parameters were set by experiments. The aim was to generate a database of n-grams (for $n = 1, \ldots, 12$) with a plausible coverage on texts but at the same time with a reasonable size. We always took case-sensitive words, numbers, and punctuation as tokens.

The predictive writing server-side script works basically in two modes:

1. If the input ends with a space, it suggests the following words, i.e. it returns the first 10 most frequent n-grams. The input is truncated to last 12 tokens and compared to the n-gram database. We strongly prefer longer n-grams, so the output is sorted by n-gram size and then by the n-gram score.
2. If the input does not end with a space, it suggests possible endings of the last word. The calculation is based on previous 11 tokens and the unfinished token.

The server-side script functionality is quite simple but for performance reasons, the n-grams were stored in finite state automata (FSA) using the `fsa` package[4].

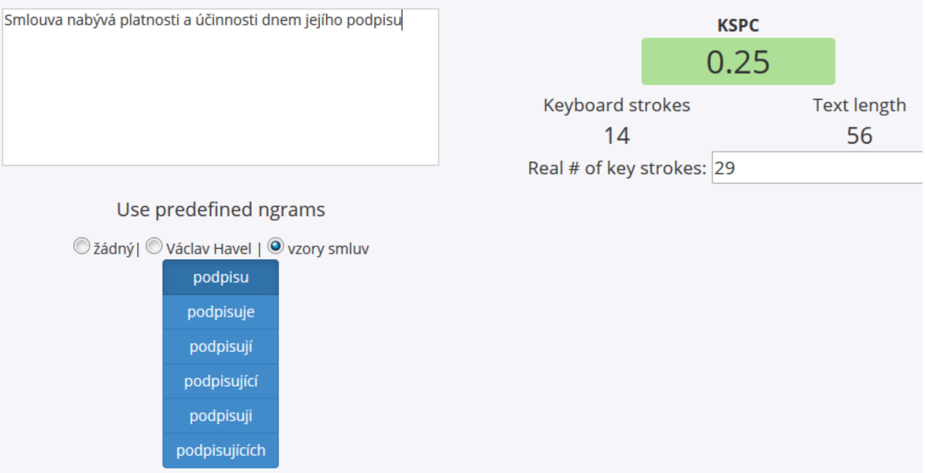
[2] In Nov 2014, it contained 5,069,447,935 tokens and 4,175,089,440 words, see `https://ske.fi.muni.cz/auth/corpora/`

[3] All tools are documented in the Sketch Engine Project Wiki: `http://www.sketchengine.co.uk/documentation/wiki/SkE/NGrams.`

[4] `http://galaxy.eti.pg.gda.pl/katedry/kiw/pracownicy/Jan.Daciuk/personal/fsa.html`

3.2 Client-side Design

The client side provides a text area for writing and selectbox with suggestions. Unlike touchscreens, the selectbox is controlled by up/down arrows and the tab key for selecting the desired word. We reduced the number of suggestions to 6 since users do not usually find it productive to read more of them. A screentshot is presented in Figure 1. The client side also trims spaces before punctuation.



Smlouva nabývá platnosti a účinnosti dnem jejího podpisu

KSPC

0.25

Keyboard strokes: 14

Text length: 56

Real # of key strokes: 29

Use predefined ngrams

○ žádný| ○ Václav Havel | ● vzory smluv

Fig. 1: Screenshot of the basic client

We strongly benefit from asynchronous JavaScript (AJAX) in order to call the server-side script after each key stroke. This procedure is quite demanding on the server-side script performance.

3.3 Extension to custom-built model

Our next goal was to improve the system tailoring it to the individual users or domains. This was done by allowing the user to upload a file with user specific (or domain specific) texts and then adjusting the system so it suggests primarily the n-grams taken from user files: we call these n-grams the *custom-built model*.

3.4 Discussion on the vocabulary size

Czech is a highly inflected language, thus it has much more word forms than e.g. English. The size of the n-gram vocabulary has to be considerably bigger. For example, [6] argue that German corpora have to be 4 times larger than English ones in order to keep the same theoretical minimum error rate in speech recognition. For Hungarian, the corpus size must be 20 times bigger compared to an English corpus.

From the czTenTen corpus, we extracted n-grams for $n = 1, ..., 12$. Table 1 shows the number of n-grams for respective n.

Table 1: Number of n-grams with respect to different n

n	number of n-grams
1	8,312,152
2	68,217,654
3	38,781,821
4	22,163,068
5	8,554,953
6	2,798,732
7	849,500
8	267,789
9	100,753
10	50,674
11	31,152
12	21,140
total	150,149,388

3.5 Building custom language models

To create a file with user n-grams, we first tokenize the user file which is expected to be in a plain text format. We use the `unitok` tool[5]. The corpus data are already tokenized so they do not have to go through this process. To find unigrams and their frequency distribution, we used a bash pipeline:

```
cat inputfilename | /corpora/programy/unitok.py -n | sort \
| uniq -c | sort -r -n | awk '{print $2 ":" $1}' > unigrams
```

This command will simply sum the number of occurrences of each word.

3.6 N-gram weighting

To find longer n-grams, we implemented a function which used the above mentioned `unitok` tool for tokenization and functions from the python nltk toolkit[6] for creating the n-grams and counting the number of their occurrences. Since we do not expect the user data to be very large, we decided to only count n-grams of the length of 2 to 4 tokens. Longer n-grams are not likely to occur more than once in a short text. Also, we did not want their frequency

[5] `https://www.sketchengine.co.uk/documentation/wiki/Website/LanguageResourcesAndTools`

[6] `http://www.nltk.org`

distribution to be the only criteria for sorting because longer n-grams are much less frequent than short ones but at the same time they are much more interesting for our purposes.

We decided to count the n-gram score according to the following formula:

$$Score = FrequencyDistribution \times n^4,$$

where n is the number of tokens in the n-gram. Then we sort the n-grams according to this score.

User n-grams are stored in text files, the server-side script uses the `sgrep` utility[7] for binary search in the text files.

4 Evaluation

Users write the text in a text area and control the prediction via the up/down arrows and the tab key. If they see the desired word in the selection box, they choose it using the arrow keys and then select the word by the tab key. We evaluated the application on several texts measuring both the letter KSPC and the real KSPC (see Section 2). Letter KSPC reflects only the performance of the language model while real KSPC also reflects the input method implementation. Corrections, deletions, and clipboard operations were not comprised in none of the measures.

4.1 Experiment

We measured KPSC on four texts that are not present in the corpus: The average letter KPSC was 0.64 and the real KSPC was 0.77.

Afterwards, we measured the KPSC according to a particular writing style. We used two completely different custom-built models: contract templates and Václav Havel's speeches. From the former source, we obtained 20k unigrams and 386k n-grams for $n = 2, 3, 4$. From the latter source, we obtained 41k unigrams and over milion n-grams for $n = 2, 3, 4$.

We then typed in a paragraph from a mortgage contract (480 characters without spaces), a contract of sale (362 characters) and a part of Václav Havel's speech (524 characters) and a paragraph from Václav Havel's essay (470 characters) with the use of the custom-built models and without them (so the tool is only using data from the general corpus). The paragraphs are long enough so the KSPC is not much influenced by a few out-of-vocabulary words. The results are shown in Table 2.

5 Results and Observations

After uploading the user data, both letter KSPC and real KSPC improved slightly. The measured KPSC shows that the tool is usable for writing arbitrary

[7] `http://sgrep.sourceforge.net/`

Table 2: Resulting KSPC on four texts

text type	mortgage contract	contract of sale	H. speech	H. essay
letter KSPC	0.63	0.55	0.68	0.71
letter KSPC with custom-built model	0.62	0.54	0.67	0.70
real KSPC	0.77	0.64	0.81	0.84
real KSPC with custom-built model	0.70	0.62	0.81	0.77

texts in Czech. Our system is comparable with other prediction systems, e.g. WordTree [7] which reports KSPC = 0.71. Due to the n-gram resource – the web corpus – it can contain non-standard n-grams, thus it is less suitable for spell checking.

The tool is more successful with contracts than Václav Havel's texts. Legal texts have much more predictable sentence structure, many commonly used phrases (resulting in n-grams with higher scores) and a vocabulary where there are lot of words used very often and fewer words are used rarely. This is common with NLP tools which are often more successful with expert domains than general texts.

With Václav Havel's texts the custom-built model improves the result as a whole but sometimes it actually make the result worse. For example, without custom-built model, after typing "já" (the pronoun *I*), the system suggests "bych" (*would*) because the strongest n-gram starting with "já" is "já bych chtěl" (*I would like*). With model built from Havel's texts, the prediction system suggests "se", as Havel's strongest n-gram is "já se domnívám" (*I assume*). However, if we type only "D" without any custom-built model, the system suggests "Dobrý" (*Good*) at the third place while model built from Havel's texts, it does not suggest "Dobrý" at all. Therefore, it will not help when a speech starts with (very common greeting) "Dobrý den" (*Hello* but literally *Good day*).

The program often suggests a word with the right base but a wrong ending. This is due to Czech being a highly inflected language. At the same time, the sentence parts with obligatory agreements do not need to be close.

6 Conclusion and Future Work

We have built a prototypical system for predictive writing and evaluated it on Czech. While predictive writing seems to be the domain of mobile devices, we found several benefits of predictive writing arising from the reduction of number of key strokes: typing speed, correct spelling, natural collocations. These benefits can be arguable and have to be measured on real-world texts. We extended the n-gram model calculated from a general corpus by user specific or domain specific texts. We proved that in some domains, predictive writing with the appropriate custom-built model can be even more effective.

Predictive writing has many applications and potential users such as motor impaired persons, users with dysgraphia, foreigners learning Czech, and touchscreen users. For this reason, we plan to improve the system and to measure its usefulness (expressed by not only the KSPC but also typing speed and number of errors) on real-world texts.

For near future, we plan to clear the n-grams in order to exclude n-grams with undoubtedly incorrect Czech and spelling errors. At the same time, we expect the KSPC decrease when the system offers more than one next word.

Other improvements comprise shift from n-gram to grammar-based prediction and learning from the user input.

Acknowledgements This work has been partly supported by the Masaryk University within the project *Čeština v jednotě synchronie a diachronie – 2014* (MUNI/A/0792/2013).

References

1. Klarlund, N., Riley, M.: Word n-grams for cluster keyboards. In: Proceedings of the 2003 EACL Workshop on Language Modeling for Text Entry Methods. TextEntry ’03, Stroudsburg, PA, USA, Association for Computational Linguistics (2003) 51–58
2. Hasegawa, A., Yamazumi, T., Hasegawa, S., Miyao, M.: Evaluating the input of characters using software keyboards in a mobile learning environment: A comparison between software touchpanel devices and hardware keyboards. In: IEEE International Conference on Wireless, Mobile, and Ubiquitous Technology in Education. (2012) 214–217
3. MacKenzie, I.: KSPC (keystrokes per character) as a characteristic of text entry techniques. In Paternò, F., ed.: Human Computer Interaction with Mobile Devices. Volume 2411 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2002) 195–210
4. Matiasek, J., Baroni, M., Trost, H.: FASTY - a multi-lingual approach to text prediction. In Miesenberger, K., Klaus, J., Zagler, W., eds.: Computers Helping People with Special Needs. Volume 2398 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2002) 243–250
5. Suchomel, V., Pomikálek, J.: Efficient web crawling for large text corpora. In Kilgarriff, A., Sharoff, S., eds.: Proceedings of the seventh Web as Corpus Workshop (WAC7), Lyon (2012) 39–43
6. Németh, G., Zainkó, C.: Word unit based multilingual comparative analysis of text corpora. In Dalsgaard, P., Lindberg, B., Benner, H., Tan, Z.H., eds.: INTERSPEECH, ISCA (2001) 2035–2038
7. Badr, G., Raynal, M.: WordTree: Results of a word prediction system presented thanks to a tree. In Stephanidis, C., ed.: Universal Access in Human-Computer Interaction. Applications and Services. Volume 5616 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2009) 463–471

One System to Solve Them All

Jan Rygl

NLP Centre, Faculty of Informatics, Masaryk University
Czech Republic
rygl@fi.muni.cz

Abstract. People are daily confronted with hundreds of situations in which they could use the knowledge of stylometry. In this paper, I propose a universal system to solve these situations using stylometry features, machine learning techniques and nature language processing tools. The proposed tool can help translation companies to recognize machine translation falsely submitted as a work of a human expert; identify school essays not written by the underwritten student; or cluster product reviews by authors and merge user reviews written by one author using multiple accounts.

All examples above use same techniques and procedures to solve the problem, therefore it is preferred to merge algorithms and implementation of these tasks to a single framework.

Keywords: stylometry, machine learning

1 Introduction

People are daily confronted with hundreds of situations in which they could use the knowledge of stylometry. I will mention several pressing problems:

Purchase of school essays during educational process: With the expansion of the Internet in the majority of households, the number of specialized web pages offering to order essays and diploma theses increased rapidly. If the submitted work was published on the Internet, the plagiarism methods can detect a fraud. Otherwise, stylometry techniques are needed to expose falsely signed works: The style of previous author's works is compared to the style of the submitted work. If the style is different enough that it exceeds the limit defined for the diversity of one author, the system will notify evaluators.

Registering using a false age or gender in dating advertisements; on discussion forums; or in Internet chats: Deception detection is the task of automatically classifying a text as being either truthful or deceptive according to the identity of author such as gender or age. In online social network communities it is easy to provide a false name, age, gender and location in order to hide a true identity, providing criminals such as pedophiles with new possibilities to harm people. Checking user profiles on the basis of text analysis can detect false profiles and flag them for monitoring [6].

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2014, pp. 19–26, 2014. © Tribun EU 2014

Machine translation submitted as human expert translation: Translation companies hire human experts (translators) to translate texts. For some of the less frequented languages it is difficult to verify the quality of the translation, therefore human experts can be tempted to use machine translation tools to complete their tasks automatically. Stylometry techniques can distinguish between automatically translated text and the text translated by a human expert.

False product reviews: During the last five years, the volume of Internet advertising doubled in Czech Republic [11]. Internet shoppers are influenced by product reviews. The share of user reviews increases at the expense of the share of professional reviews. The number of products rises faster than is the capacity of magazines aimed at user reviews. This situation leads to the fact that some companies are guilty of unfair trade practices and creates fake product reviews: positive ones to improve the rating of their goods, and negative ones to harm their competitors [9]. We can fight false reviews by recovering true authorship of reviews; cluster user accounts by their true author; and detect automatically generated reviews.

2 Stylometry

Author's style is defined as a set of measurable text features according to stylostatisticians [8]. These features are called style markers. Word-length frequencies were used as the first style markers to detect an authorship of documents. T. C. Mendenhall discovered that word-length frequency distribution tends to be consistent for one author and differs for different authors (1887, [5]).

Style markers can be divided into categories, which can be defined by properties of texts that are used, or by tools needed to extract information.

Usually, following tool categories are used to implement stylometry techniques (examples of Czech tools are given):

1. Text cleaning (boiler-plate removal, HTML removal, etc.)
2. Language detection
3. Encoding detection (Chared[1])
4. Text tokenization
5. Morphology analysis (Majka [10])
6. Syntactic analysis (SET [4])
7. Semantic analysis (entity detection, abbrevation expansion, etc.)

The number of categories based on extracted information is still growing, therefore only a few predominant examples are listed:

1. Wordclass n-grams
2. Morphology tags n-grams
3. Word-length and sentence-length distribution
4. Typography errors

[1] http://nlp.fi.muni.cz/projects/chared/

5. Punctuation usage
6. Subtrees from a tree generated by syntactic analysis

The quality and the utility of style markers depend on the type of problem. Different document lengths and tasks require different style markers, therefore it is recommended to experimentally select a subset of style markers and not to use them all [7].

3 Machine learning

Machine learning techniques work with data instances. Each instance is an n-tuple of features, each feature represents one style marker.

The instances are separated into two groups. Training group is labeled and contains information about author, gender, age, etc., depending on the scenario. Training instances are used to create a machine learning classifier. The classifier is given unlabeled test instances and predicts labels. The features are usually rational numbers, which are automatically normalized to a range $\langle 0,1 \rangle$ or $\langle -1,1 \rangle$.

To solve the problems using stylometry techniques, two Support Vector Machines methods are recommended [3]:

- SVM implementation LIBSVM [1]
- Linear SVM implementation LIBLINEAR [2]

The selected SVM based techniques have several parameters which should be tuned for each data set. Grid search and other optimization techniques are used to find the best parameters for learning data set.

Depending on what is used as a data instance, we can distinguish two approaches 3.1 and 3.2.

3.1 One model per label approach

For each label (labels can be ages, genders, author names, types of translations), one machine learning model is trained. Each model for a label L classifies whether a given document should be given the label L. The n most probable labels are selected for each document ($n = 1$ for a majority of tasks).

The advantage of this approach is that it supports tasks with multilabeled instances (e.g. document written by more authors). The disadvantage is that it requires training instances for each label, therefore this approach cannot predict labels for test instances with unseen labels.

The data instance is an n-tuple of style markers of one document.

3.2 Similarity approach

Similarity approach is used to compare two documents and predict the similarity between them. Given two documents A and B, style-marker n-tuples $s(A)$

and $s(B)$ are extracted. The inverse absolute difference of style-marker n-tuples (similarity) is counted:

$$1 - |s(A)[1] - s(B)[1]|, 1 - |s(A)[2] - s(B)[2]|, \ldots, 1 - |s(A)[n] - s(B)[n]$$

where $s(A)[i]$ is i-th item of $s(A)$ and $s(B)[j]$ is j-th item of $s(B)$.

The data instance is a similarity n-tuple of two style markers. This approach can compare whether two documents have the same label even if the label is not present in training instances.

4 One system

Most of the previously mentioned techniques are common for algorithms solving stylometric problems. Therefore, I proposed a system schema which can be used to solve all tasks with minimal effort. The schema consists of following parts (training a model):

1. Annotating (each doment is given a label)
2. Document processors (documents are cleaned and expanded to a collection of extracted information)
   (a) text cleaning (remove boiler-plae, HTML, ...)
   (b) language detection
   (c) charset detection
   (d) tokenization
   (e) morphology analysis
   (f) syntactic analysis
   (g) semantic analysis (abbreviations, entities, ...)
3. Style extraction (expanded documents are converted to feature n-tuples, where n is the number of style markers)
4. Similarity extraction (if we want to solve a task using a similarity approach, feature n tuples of selected document pairs are compared and similarity n-tuples are counted)
5. Machine learning – training a model (each n-tuple has a label)
   (a) feature selection (select the best combination of style markers)
   (b) machine learning parameters selection
   (c) model creation

The schema for classification consists of the following parts (see Figure 1):

1. Document processors (see a previous List)
2. Style extraction (expanded documents are converted to feature n-tuples, where n is the number of style markers)
3. Similarity extraction (if we want to solve a task using thea similarity approach, feature n tuples of selected document pairs are compared and similarity n-tuples are counted
4. Machine learning classification (each n-tuple is given a label)

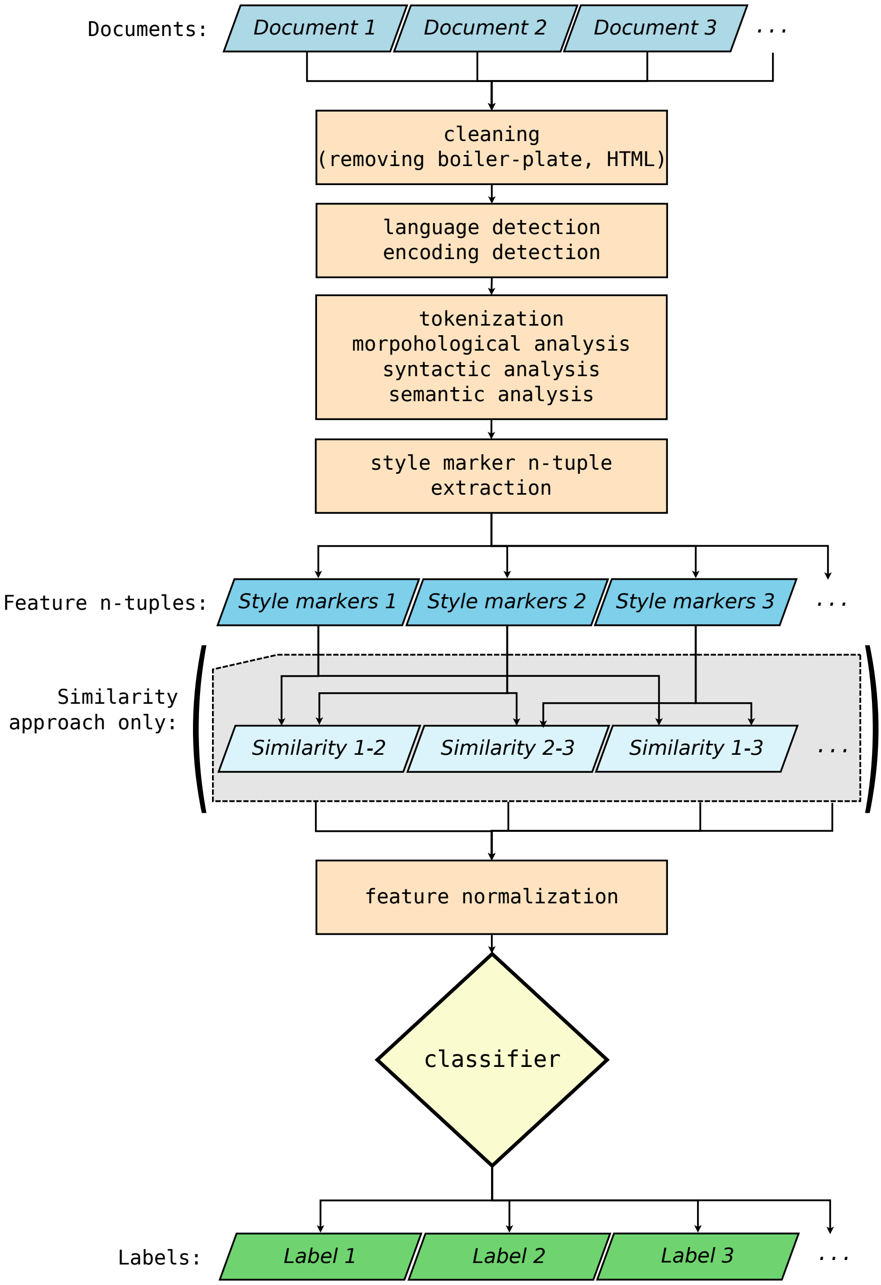


Fig. 1: System schema: document classification

4.1 Authorship of school essays

Input: 2-tuples (author, document).

Style extraction: Select style markers based on genres of documents.

Machine learning: Train a model with two labels ([author's name], other) for each author. Author's documents are used as instances with the label [author's name], documents of other authors have the label other.

Classification: For each author's model, estimate a probability of each label. Check whether document signed by author A has the label A with the probability higher than probabilities of all other labels except the other label. If author's probability is lower than some probability of other author, notify evaluators.

4.2 False product reviews

Input: 2-tuples (author, document).

Style extraction: Select style markers suitable for short texts.

Similarity extraction: Compare each two documents and extract similarities between them.

Machine learning: Train one model. Comparison of two documents of one author are given a label same_authors, pairs of documents signed by different authors are used as instances with a label different_authors.

Classification: Check whether pairs consisting of documents from two different authors are labeled as same_authors. If more than one document pair of two authors is classified with the same authorship, consider merging these authors.

4.3 Registering using a false age in dating advertisements

Input: 2-tuples (age, document).

Style extraction: Use all style markers.

Machine learning: Divide ages into several groups, each group is represented by one label (e.g. gradeschooler, teen, young_adult). Train one model using these labels.

Classification: Check whether the document is classified as the same label as the document is annotated. If the label does not match, notify system administrators.

4.4 Machine translation submitted as human expert translation

Input: 2-tuples `(source, document)`.

Style extraction: Select style markers based on genres of documents.

Machine learning: Train one model. Machine translation instances are labeled as `machine_translation`, documents translated by human experts have the `human_translation` label.

Classification: Check if the submitted translation is given the `human_translation` label. If the label does not match, evaluate the translation by another human expert.

5 Conclusions and future work

I plan to implement a multilingual system according to the proposed schema. The system will use the state of the art libraries for machine learning techniques and text processing, and wide range of stylometric features. Once implemented, all scenarios mentioned in this paper will be tested using this system and the results will be published.

Acknowledgements This work has been partly supported by the Ministry of Education of CR within the LINDAT-Clarin project LM2010013.

References

1. Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. URL: `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.
2. Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A Library for Large Linear Classification. *J. Mach. Learn. Res.*, 9:1871–1874, June 2008.
3. Moshe Koppel, Jonathan Schler, and Shlomo Argamon. Computational methods in authorship attribution. *J. Am. Soc. Inf. Sci. Technol.*, 60:9–26, January 2009.
4. Vojtěch Kovář, Aleš Horák, and Miloš Jakubíček. Syntactic analysis using finite patterns: A new parsing system for Czech. In *Human Language Technology. Challenges for Computer Science and Linguistics*, Lecture Notes in Computer Science, pages 161–171, Berlin, 2011. Springer.
5. T. C. Mendenhall. The characteristic curves of composition. *The Popular Science*, 11:237–246, 1887.
6. Claudia Peersman, Walter Daelemans, and Leona Van Vaerenbergh. Predicting age and gender in online social networks.
7. Jan Rygl. Automatic Adaptation of Author's Stylometric Features to Document Types. In *Text, Speech and Dialogue - 17th International Conference. 8655.*, pages 53–61. Brno: Springer, 2014., 2014.

8. Efstathios Stamatatos, Nikos Fakotakis, and George Kokkinakis. Automatic text categorization in terms of genre and author. *Computational Linguistics*, 26(4):471–495, Dec 2000.
9. Ben Verhoeven and Walter Daelemans. Clips stylometry investigation (csi) corpus: A dutch corpus for the detection of age, gender, personality, sentiment and deception in text. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 3081–3085, Reykjavik, Iceland, May 2014. European Language Resources Association (ELRA). ACL Anthology Identifier: L14-1001.
10. Pavel Šmerk. Fast morphological analysis of Czech. In *Proceedings of Third Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 13–16, Brno, 2009. Tribun EU.
11. Association for Internet Advertising. ppm factum, Admosphere, Kantar Media, February 2014. URL `http://www.spir.cz/en/internet-advertising-exceeded-czk-13-billion-last-year-doubling-over-last-five-years`

.

Improving Coverage of Translation Memories with Language Modelling

Vít Baisa, Josef Bušta, and Aleš Horák

Natural Language Processing Centre
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
{xbaisa,xbusta1,hales}@fi.muni.cz

Abstract. In this paper, we describe and evaluate current improvements to methods for enlarging translation memories. In comparison with the previous results in 2013, we have achieved improvement in coverage by almost 35 percentage points on the same test data. The basic subsegment splitting of the translation pairs is done using Moses and (M)GIZA++ tools, which provide the subsegment translation probabilities. The obtained phrases are then combined with subsegment combination techniques and filtered by large target language models.

Keywords: translation memory, CAT, segment, subsegment leveraging, partial translation, Moses, GIZA++, word matrix, METEOR, MemoQ, language model

1 Introduction

Computer-aided translation (CAT) is becoming more and more popular—with the state-of-the-art technologies such as subsegment leveraging, machine translation, or automatic terminology extraction, the translation process is faster and easier than ever before.

CAT systems depend on translation memories: manually built databases of aligned source and target segments (phrases, sentences, paragraphs). They can be considered as parallel corpora of very high-quality (since they are prepared by professional translators) but of quite small size and coverage of new documents.

We describe current improvements of the methods for expanding translation memories which have been described in the previous paper [1]. The goal of these methods is to increase new document coverage of a translation memory preserving its high translational precision.

There is also a commercial aspect of this research: the coverage analyses provided by CAT systems are usually used for estimating the amount of work needed for translating a given document (i.e. the price of the translation work). The higher number of segments which can be pre-translated automatically, the lower is the price of the translation work. That is why the translation (and localization) companies aim at the highest coverage of their resources.

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2014, pp. 27–34, 2014. © Tribun EU 2014

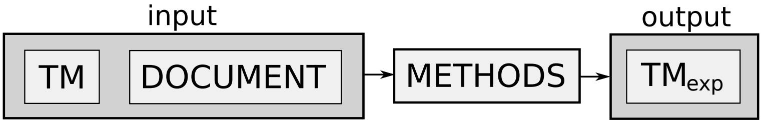


Fig. 1: Schema of the basic work flow for TM$_{exp}$.

2 Previous and Related Work

In the previous paper [1], we have proposed several methods for enlarging translation memories and provided an evaluation for one of them. In this paper, we describe the improvements of the methods and evaluate all of them on both the original data used in the previous paper and also on a new data, Directorate-General for Translation or DGT[1] [2] translation memory released recently by the European Commission. For related work refer to [1].

3 Subsegment Processing Methods

In this section, we present the changes and improvements to the previous paper [1] and a detailed description of the implemented techniques.

The input for our methods is a translation memory and a document. We want to enlarge the TM (the expanded TM is denoted TM$_{exp}$) to cover more segments in the document and preserve the quality of the translations, see the Figure 1.

3.1 Method A: Subsegment Generation

Subsegments and the corresponding translations are generated using Moses [3] tool directly from the TM, no additional data is used. The word alignment is based on MGIZA++ [4] (parallel version of GIZA++ [5]) and the default Moses heuristic *grow-diag-final*.[2] The next steps are phrase extraction and scoring [3]. The corresponding partially expanded TM is denoted as TM^{sub}. The output from subsegment generation has the following format:

Subsegment	Translation	Probabilities	Alignment points
nejlepší uhlí	best coal	0.158, 0.142, 0.158, 0.69	0-0 1-1

The probabilities are *inverse phrase translation probability*, *inverse lexical weighting*, *direct phrase translation probability* and *direct lexical weighting* obtained directly from the Moses procedures. These probabilities are used to select the best

[1] `https://ec.europa.eu/jrc/en/language-technologies/dgt-translation-memory`

[2] `http://www.statmt.org/moses/?n=FactoredTraining.AlignWords`

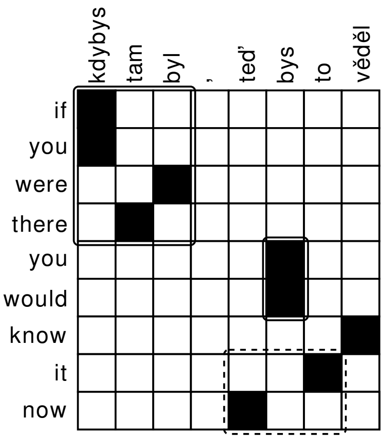


Fig. 2: Word matrix for two aligned sentences / segments.

translations in case there are many translations for a subsegment. Alternative translations for a subsegment are combined from different aligned pairs in the TM. Typically, short subsegments have many translations.

The alignment points determine the word alignment between subsegment and its translation, i.e. *0-0 1-1* means that the first word *"nejlepší"* from the source language is translated to the first word in the translation *"best"* and the second word *"uhlí"* to the second word *"coal."* These points give us an important information about the subsegment translation: 1) empty alignment, 2) one-to-many alignment, and 3) opposite orientation.

In Figure 2 the empty alignment is represented by an empty line or an empty row, the one-to-many alignment by a sequence of adjacent squares in a row or in a column and the opposite orientation by a sequence of neighbouring squares on the secondary diagonal. The alignments are used to determine correct positions in the subsegments translations.

3.2 Method B: Subsegment Combination

The subsegment translation pairs obtained by the method A are used as a pool of candidate subsegments used in the next method to generate longer subsegments. In an ideal case, to generate a new translation pair covering a whole, originally uncovered, segment in the input document – so called *100 % match*.

Currently, the sub-methods *join* and *substitute* are proposed for subsegment combinations, each of them in an *overlapping* and *non-overlapping* variant:

1. JOIN: new segments are built by concatenating two segments from TM^{sub}, denoted TM^{J}.
   (a) JOIN^{O}: joined subsegments overlap in a segment from the document, denoted TM^{OJ}.

Table 1: SUBSTITUTE^{O}, example for Czech → English

new subsegment	Provozovatelé musí dodržovat zvláštní pravidla pro výzkumné
its translation	Operators shall comply with the special rules on research
from subsegments	Provozovatelé **musí** vytvářet **zvláštní** pravidla pro výzkumné | **musí** dodržovat **zvláštní**
their translations	Operators **shall** create **the special** rules on research | **shall** comply with **the special**

   (b) JOIN^{N}: joined subsegments neighbour in a segment from the document, denoted TM^{NJ}.

2. SUBSTITUTE: new segments can be created by replacing a part of one segment with another subsegment from TM^{sub}, denoted TM^{S}.
   (a) SUBSTITUTE^{O}: the gap in the first segment is covered with an overlap with the second subsegment, see the example in Table 1, denoted TM^{OS}.
   (b) SUBSTITUTE^{N}: the second subsegment is inserted into the gap in the first segment, denoted TM^{NS}.

During the subsegment non-overlapping combination, any two subsegments are combined regardless the fluency and the context. That is why we need to evaluate the quality of the combination. For the quality measurement, we have trained a language model using KenLM [6] tool on first 50 million sentences from enTenTen [7] with model order set to 5.

The translation quality of the SUBSTITUTE operation can be improved by substituting a particular part-of-speech (noun, adjective, ...) for the same part-of-speech or a noun phrase for a noun phrase.

Algorithm 1: JOIN subsegments

```
Data: Segment $S$ from document; List $I$ of indexes $(i,\ j)$ of subsegments occurring
      in $S$ sorted in decreasing order by the difference of $j - i$
Result: $R$
while $I \neq \emptyset$ do
    $(i,j) \longleftarrow$ First($I$);
    $I \longleftarrow I - (i,j)$;
    $T \longleftarrow \emptyset$;
    for $(k,\ l) \in I$ do
        if $(k < i \wedge l + 1 \geq i \wedge j > l) \vee (i < k \wedge j + 1 \geq k \wedge l > j)$ then
            $T \longleftarrow T +$ (Min $(k,i)$,Max $(l,j)$);
            $R \longleftarrow R +$ (Min $(k,i)$,Max $(l,j)$);
            if (Min $(k,i)$,Max $(l,j)$) $=$ (0,Length $(S)$) then
                return $R$;
    $I \longleftarrow T + I$;
return $R$;
```

In [1], the operation JOIN was implemented just for non-overlapping subsegments and as a concatenation of any two subsegments. In this paper, we present an improved Algorithm 1. The algorithm works with indexes which represent the subsegment positions in the tokenized segment from the input document. The processing starts with the biggest subsegment in the segment and then tries to join it with other subsegments. If it succeeds, the new subsegment is appended to temporary list T. After all other subsegments are processed, T is prepended to I and the algorithm starts with a new subsegment created from the two longest subsegments. If it does not succeed, the next subsegment in the order is processed. The algorithm 1 prefers to join longer subsegments. In each iteration it generates new (longer) subsegments and it discards one processed subsegment. See Section 4 for the evaluation of this new approach.

4 Evaluation

For the evaluation of the current implementation of the TM-expanding methods, we have used the same translation memory TM^{s} and the same example document D^{s} as in [1]. Both data files have been provided by one of the biggest Czech translation companies.

Table 2: MemoQ analysis for TM^{s}.

	TM				TM^{sub}				TM^{NS}			
Match	**Seg**	**wrds**	**chars**	%	**Seg**	**wrds**	**chars**	%	**Seg**	**wrds**	**chars**	%
100%	23	128	813	0.4	165	178	611	0.51	0	0	0	0
95–99%	45	185	1,130	0.5	193	245	1,578	0.7	20	43	273	0.12
85–94%	4	21	155	0.1	19	50	325	0.14	18	78	451	0.22
75–84%	42	208	1,305	0.6	96	310	1,888	0.88	129	436	2,677	1.24
50–74%	462	1,689	10,293	4.8	789	4,543	27,999	12.93	1681	12,522	75,108	35.65
$\geq$ 75%	114	542	3403	1.6	473	783	4,402	2.23	167	557	3,401	1.58
any	576	2,231	13,696	6.4	1,262	5,326	32,401	15.16	1,848	13,079	78,509	37.23

	TM^{OJ}				TM^{NJ}				TM^{all}			
Match	**Seg**	**wrds**	**chars**	%	**Seg**	**wrds**	**chars**	%	**Seg**	**wrds**	**chars**	%
100%	6	23	106	0.07	4	19	101	0.05	182	302	1360	**0.86**
95–99%	11	60	310	0.17	13	87	466	0.25	232	465	2,858	1.32
85–94%	5	33	217	0.09	17	149	892	0.42	41	221	1,382	0.63
75–84%	68	314	1,809	0.89	110	881	5,022	2.51	265	1,475	8,655	4.2
50–74%	1,153	7,667	45,641	21.83	1,354	11,997	70,730	34.15	1,507	15,324	92,158	43.62
$\geq$ 75%	90	430	2,442	1.22	144	1,136	6,481	3.23	720	2,463	14,255	7.01
any	1,243	8,097	48,083	23.05	1,498	13,133	77,211	37.38	2,227	17,787	106,413	**50.63**

Table 3: MemoQ analysis for DGT-TM.

	TM				TM^{sub}				TM^{NS}			
Match	**Seg**	**wrds**	**chars**	**%**	**Seg**	**wrds**	**chars**	**%**	**Seg**	**wrds**	**chars**	**%**
100%	31	59	639	0.03	276	457	2,666	0.25	58	260	953	0.45
95–99%	198	546	1,941	0.30	225	446	1,998	0.24	206	827	2,992	0.45
85–94%	43	169	986	0.09	208	971	4,205	0.53	94	492	2,187	0.27
75–84%	357	1,745	8,021	0.96	386	1,714	9,115	0.94	287	1,492	7,102	0.82
50–74%	2,580	20,778	126,273	11.37	2,907	22,736	141,526	12.45	3,348	29,549	182,667	16.18
$\geq$ 75%	629	2,519	11,587	1.38	1,095	3,588	17,984	1.96	645	3,071	13,234	1.99
any	3,209	23,297	137,860	12.75	4,002	26,324	159,510	14.41	3,993	32,620	195,901	17.86

	TM^{OJ}				TM^{NJ}				TM^{all}			
Match	**Seg**	**wrds**	**chars**	**%**	**Seg**	**wrds**	**chars**	**%**	**Seg**	**wrds**	**chars**	**%**
100%	38	187	764	0.10	29	161	683	0.09	358	838	4,172	**0.46**
95–99%	195	770	2,752	0.42	69	247	769	0.14	338	990	4,282	0.54
85–94%	124	695	3,198	0.38	203	1,107	4,892	0.61	133	666	3,750	0.36
75–84%	256	1,634	7,764	0.89	287	2,133	10,331	1.17	537	3,231	17,340	1.77
50–74%	3,220	32,325	200,667	17.70	3,673	47,715	298,031	26.12	4,183	53,791	343,699	29.45
$\geq$ 75%	613	3,286	14,478	1.79	588	3,648	16,675	2.01	1,366	5,725	29,544	3.13
any	3,833	35,611	215,145	19.49	4,261	51,363	314,706	28.13	5,549	59,516	373,243	**32.58**

The evaluation results have been obtained directly from the pre-translation analysis of the MemoQ[3] system. The statistics express how many segments from the document D^s can be translated automatically using the TM-expanding methods. The automatic translation is done on the segment level and even on lower levels of subsegments. The partial matches are expressed as the match percentages in the table. The 100% match corresponds to the situation when a whole segment from D^s can be translated using a segment from the respective translation memory (either the original one or a memory obtained by each particular sub-method). Translations of shorter parts of the segment are then matches lower than 100%.

The columns in Tables 2 and 3 are: **Match**: type of match between TM and D^s, **Seg**: number of segments identified in D^s, **wrds**: number of source words which are covered (translatable) by TM, **chars**: number of source characters, and percent sign: percentage of coverage for the type of match in the first column. In the evaluation process, we have first tested the translation on a document with 4,563 segments (35,142 words and 211,407 characters), see Table 2.

For an independent comparison, we also present our results for DGT translation memory [2]. For the evaluation using DGT we have used 330,626 pairs from 2014 release and evaluated it on 10,000 randomly chosen segments from the same release. Duplicate pairs were removed before evaluation. See Table 3 for the results.

[3] `http://kilgray.com/products/memoq`

Table 4: Analysis of dependence between subsegment length and the coverage of the document.

	TM^{s}					DGT-MT				
Length	TM^{sub}	TM^{OJ}	TM^{NJ}	TM^{NS}	TM^{all}	TM^{sub}	TM^{OJ}	TM^{NJ}	TM^{NS}	TM^{all}
≥ 1	85%	11%	15%	25%	85%	95%	57%	71%	65%	95%
≥ 2	35%	11%	15%	25%	44%	78%	57%	71%	65%	85%
≥ 3	7%	11%	15%	25%	32%	53%	57%	71%	65%	82%
≥ 4	1%	5%	15%	9%	16%	35%	52%	71%	53%	74%
≥ 5	0%	2%	8%	2%	7%	23%	45%	66%	38%	65%

Table 5: Translation quality (METEOR score) for 100% matches.

	TM^{s}					DGT-MT				
feature	TM^{sub}	TM^{OJ}	TM^{NJ}	TM^{NS}	TM^{all}	TM^{sub}	TM^{OJ}	TM^{NJ}	TM^{NS}	TM^{all}
precision	0.60	0.63	0.70	0.66	0.61	0.76	0.93	0.91	0.81	0.80
recall	0.67	0.74	0.74	0.71	0.68	0.78	0.86	0.88	0.85	0.81
f1	0.64	0.68	0.72	0.68	0.64	0.77	0.89	0.89	0.83	0.81
METEOR score	0.31	0.37	**0.38**	**0.38**	0.31	0.40	0.50	**0.51**	0.45	0.43

We have also counted the coverage of the document considering the length of subsegments, see Table 4. Notice that longer subsegments are created by subsegment combination.

The METEOR [8] metric was used to evaluate quality (precision) of the proposed translated segments. We provide statistics for all implemented methods on both test data sets, see Table 5. The METEOR evaluation metric has been proposed to evaluate MT systems, therefore it assumes that we have fully translated segments (pairs). That is why we are evaluating only 100% matches since it is not straightforward to interpret METEOR scores for partially translated candidate sentences.

We have analysed the problematic cases regarding the precision. The most common error is when subsegments are combined in the order in which they occur in the segment assuming the same text sequential order in the target language, see the Table 6. We assume, that such errors will be less frequent with a larger input translation memory, which will offers higher ration of the overlapped (contextual) segments.

Table 6: Non-overlapping JOIN error example Czech → English.

segment	Prémie na bramborový škrob
reference	Potato starch premium
new subsegment	Prémie na bramborový škrob
its translation	Premiums potato starch
from subsegments	Prémie na | bramborový škrob
their translations	Premiums | potato starch

5 Conclusions

We have shown that the originally proposed methods can be further improved and provided the evaluation which shows that the coverage of all matches has been increased by 34.5 percentage points (from 16.15% reported in [1] to 50.63%). As for the 100% matches which are the most important, the test results show an increase of 0.5 percentage points comparing the original TM and combination of both JOIN methods (from 0.4% reported earlier to 0.86%) and the coverage of $>$ 75% matches increased by 5.4% (from 1.6% to 7%).

The translational quality of the resulting new segments is kept at the high level as is shown by the METEOR score up to 0.51 for the evaluation with the translation memory by Directorate-General for Translation (DGT) of the European Commission.

Acknowledgements The work has been partly supported by the OP VaVpI project No CZ.1.05/3.1.00/10.0216.

References

1. Baisa, V., Bušta, J., Horák, A.: Expanding translation memories: Proposal and evaluation of several methods. RASLAN 2013 Recent Advances in Slavonic Natural Language Processing (2013) 71
2. Steinberger, R., Eisele, A., Klocek, S., Pilos, S., Schlüter, P.: Dgt-tm: A freely available translation memory in 22 languages. arXiv preprint arXiv:1309.5226 (2013)
3. Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., et al.: Moses: Open source toolkit for statistical machine translation. In: Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions, Association for Computational Linguistics (2007) 177–180
4. Gao, Q., Vogel, S.: Parallel implementations of word alignment tool. In: Software Engineering, Testing, and Quality Assurance for Natural Language Processing, Association for Computational Linguistics (2008) 49–57
5. Och, F.J., Ney, H.: A systematic comparison of various statistical alignment models. Computational linguistics **29**(1) (2003) 19–51
6. Heafield, K.: Kenlm: Faster and smaller language model queries. In: Proceedings of the Sixth Workshop on Statistical Machine Translation, Association for Computational Linguistics (2011) 187–197
7. Jakubíček, M., Kilgarriff, A., Kovář, V., Rychlý, P., Suchomel, V., et al.: The tenten corpus family. In: Proc. Int. Conf. on Corpus Linguistics. (2013) `http://www.sketchengine.co.uk/documentation/wiki/Corpora/enTenTen`
8. Denkowski, M., Lavie, A.: Meteor universal: Language specific translation evaluation for any target language. In: Proceedings of the EACL 2014 Workshop on Statistical Machine Translation. (2014)

Part II

Text Corpora

Optimization of Regular Expression Evaluation within the Manatee Corpus Management System

Miloš Jakubíček and Pavel Rychlý

Natural Language Processing Centre
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
{jak,pary}@fi.muni.cz

Lexical Computing Ltd.
Brighton, United Kingdom
{milos.jakubicek,pavel.rychly}@sketchengine.co.uk

Abstract.
This paper is concerned with searching large text corpora – electronic collections of texts. Often these are subject to queries specified by means of regular expressions. Such queries go beyond a simple keyword search that can be quickly evaluated using an inverted index, usually they are rather processed by third-party regular expression libraries and take significantly more time to evaluate. In this paper we present an index-based approach for optimization of regular expression evaluation that we call *n-gram prefetching*. It is based on the assumption that most regular expression queries on text corpora contain at least some fixed string portions representing clues that can be used for developing heuristics that would prune the number of potentially matching strings. The presented work has been designed and implemented within the Manatee corpus management system. We show that the proposed approach can significantly speed up regular expression processing by providing evaluation on a test set of queries executed on a number of billion-word text corpora.

Keywords: text corpus, regular expression, Manatee

1 Introduction

Text corpora represent primary data resource for testing hypotheses, providing evidence and building large scale statistical models used in various natural language processing applications such as part-of-speech tagging, parsing or machine translation.

Special database management systems (in this case corpus management systems) devised for indexing and querying large text corpora have been developed to satisfy user needs in terms of complexity of queries and related response time, such as [1,2]. These corpus management systems usually leverage the idea of inverted index (inverted text) [3] to provide fast access to all occurrences of a given word (or another attribute like lemma or tag depending on the type of annotation) in the corpus.

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2014, pp. 37–48, 2014. © Tribun EU 2014

In many cases users formulate the queries not in the form of fixed string expressions but as regular expressions. Clearly this represents a serious challenge for the corpus management systems as the first step necessary to answer such queries lies in evaluating the regular expression against the relevant lexicon used in the corpus so as to be able to retrieve matching strings from the indices. Usually, a third-party regular expression library is used for the actual matching, often providing expressive power going way beyond what regular expression offer as a classical computational model.

In this paper we present an optimization of regular expression evaluation that we call *n-gram prefetching*. The approach has been implemented within the Manatee corpus management system [4] and exploits the PCRE regular expression library [5], however we claim that it is suitable for any inverted-index-based corpus management system and it is in no way dependent on any particular regular expression library.

The structure of this papers is as follows: we first provide a brief overview of the Manatee corpus management system and its overall indexing machinery, then we present the optimization approach in detail and finally we provide and evaluation on a number of billion word corpora showing a up to 100-times speedup.

2 Manatee

Manatee is a state-of-the-art corpus management system providing facilities for efficient indexing (compiling) and searching billion-word-sized corpora [6]. Querying corpora indexed by Manatee is done using the Corpus Query Language (CQL, [7]). From a formal perspective a corpus in manatee consists of *text data* (called tokens or positions, each of which may be associated with a number of attributes such as word, tag or lemma, further referred to as positional attributes) and *text metadata* (called structures, each of which is denoting a span in the corpus such as a document, paragraph or sentence, and may be associated with arbitrary number of structure attributes, denoting e.g. the author of a document, date of creation etc.).

Every positional or structural attribute possesses following basic index structures:

- **attribute lexicon** providing efficient string↔ID mapping. Each unique attribute string value is assigned a unique numeric ID which is further used in all indices and for processing CQL queries,
- **attribute inverted (reversed) index** providing sequential access to a sorted list of occurrences (corpus positions) of a given attribute ID,
- **attribute text** storing the actual text of this corpus attribute, i.e. a sequence of attribute IDs in the order of occurrence in the corpus.

On a very abstract level evaluating a CQL query consists of mapping attribute strings given in the query to their IDs (using the lexicon index), retrieving the relevant positions from the inverted index, combining them according to the given CQL operators and displaying the results (e.g. in the form of a concordance). In the simplest case, considering a query looking for all occurrences of a single word in corpus like `[word="someword"]`, one yields the ID *n* of `someword` from the lexicon and then retrieves the sorted list of positions for *n* from the inverted index. Further CQL operations always rely on processing sorted streams (of corpus positions or attribute IDs).

In the case attribute constraints are given as regular expressions, the straightforward string-to-number mapping using the lexicon is not possible. First one needs to find out which attribute values are matching the given regular expression (by scanning the whole lexicon), then map each of the values to the respective ID and merge all the related position streams retrieved from the inverted index. Since all the streams and results must be sorted by design, before the matching has finished no results are available to the user. This exhibits a serious issue for large (i.e. billion-word-sized corpora) corpora where the lexicon size is often reaching tens of millions of values and hence the time taken to evaluate the regular expressions across the whole lexicon represents a significant slow down of query evaluation, and especially of retrieving first *n* results.

Two basic optimization have already been in place to tackle this problem:

- **any string optimization**: a regular expression matching `^(\.\*)+$` was omitted as it obviously must match the whole lexicon,
- **prefix optimization**: since the lexicon provides an index of attributes ID sorted alphabetically by the corresponding string values (representing one of the possible implementations of string↔ID mapping based on a simple binary search, see [8] for comparison), all regular expressions containing a fixed-string prefix like `re.*` or `mis.*ing` make it possible to shrink the set of possible matching strings by selecting the range of IDs with the given prefix (here `re` and `mis`, respectively).

3 n-gram prefetching

In this paper we describe a new optimization approach that we call *n-gram prefetching*. It is based on the assumption that most user queries exploiting regular expressions still contain some fixed-string portions (because they are linguistically motivated). While queries like `^.{0,3}$` are of course possible, they are very rare. To verify this hypothesis we have inspected a set of 128,406 queries which the users of Sketch Engine (a web service exploiting Manatee, see [9]) have issued to that system over the period from June to September 2014. Only 12 of them did not contain any fixed-string portions, moreover 6 of these 12 were `^.*$` and got optimized as well.

The idea of n-gram prefetching consists (on compile time) in indexing all character uni-, bi- and trigrams of every string attribute value and (on run time) in extracting such n-grams from the regular expressions and using them to constrain the number of possibly matching strings from whole lexicon to a much smaller set of IDs.

3.1 n-gram indexing

For indexing of the character n-grams we leverage the idea of *dynamic attributes* in Manatee. A positional or structural attribute in Manatee can be a so called *dynamic attribute* in which case such an attribute is automatically derived from another existing (regular or dynamic) attribute. Each dynamic attribute is assigned a dynamic function which takes the source attribute string as input (and optionally other parameters as well) and returns the new (dynamic) attribute value. Manatee contains a predefined set of dynamic functions which mostly focus on simple string manipulations [1] (such as getting a prefix or suffix of a string or its lowercase variant) and users can supply their own dynamic functions as well (in the form of Linux plugins – dynamically linked C/C++ libraries). The main benefits of a dynamic attribute are:

- **space savings in source data:** no need to make it part of the input vertical text
- **space savings in indexing:** a dynamic attribute has only the lexicon and inverted index, but no text index. The inverted index stores for each dynamic attribute ID a sorted list of source attribute IDs which map to this dynamic attribute ID (instead of storing corpus positions).
- **very limited runtime overhead:** depending on the type of operations, the overhead (slowdown) of using a dynamic attribute instead of a regular one is very small. Optionally an index providing mappings of source attribute ID to dynamic attribute ID is compiled as well, in which case the dynamic functions do not need to be executed at runtime at all (except where it is necessary to convert input user query, e.g. in case of lowercasing).

Each lexicon item (string) is processed generating all occurring uni-, bi- and trigrams as shown in Figure 1 and storing the n-gram values as a dynamic attribute of the source attribute.

```
classical  ->  c|l|a|s|s|i|c|a|l
               cl|la|as|ss|si|ic|ca|al
               cla|las|ass|ssi|sic|ica|cal
```

Fig. 1: Uni-, bi- and trigram string generation.

The resulting dynamic attribute contains a lexicon with all the n-grams found in the source attribute and provides fast access to all source attribute IDs containing a given n-gram. We prepend the caret sign (^) and append the dollar sign ($) to each string so that we generate specific n-grams occurring at beginnings and ends of words.

[1] See https://www.sketchengine.co.uk/documentation/wiki/SkE/Config/DynamicAttributes.

Table 1: Comparison of original and n-gram lexicon sizes

corpus	language	size $\times 10^9$	attribute	lexicon size	n-gram lexicon size
czTenTen12	Czech	5.126	word	18,978,703	522,745
			lemma	14,151,454	506,580
			tag	12,061	1,702
enTenTen12	English	12.968	word	27,894,538	1,880,911
			lemma	26,426,200	1,880,808
			tag	60	260
enClueWeb09		82.581	word	115,820,931	2,350,697
			lemma	110,606,268	2,296,072
			tag	60	260
jpTenTen11	Japanese	10.322	word	13,844,200	6,353,186
			lemma	13,303,479	3,766,160
			tag	53	297

3.2 n-gram matching

On runtime we parse the given regular expression and extract all occurring fixed-string uni-, bi- and trigrams (preferring longer n-gram where available and combining strings longer than 3 characters into a set of trigrams by the logical `AND` operator in CQL). Since regular expression as such can be quite complicated we exploit the ANTLR3 parser generator [10] for processing the regular expressions. We have chosen ANTLR3 because it has already been used within Manatee (for CQL and corpus configuration files parsing) and it has very powerful grammar writing formalism. The ANTLR3 lexer and parser grammar is provided in Annex 1. The output of the ANTLR3 lexing and parsing is an abstract syntax tree (AST) that is further subject to parsing by ANTLR3 using a so called tree walker which directly executes programming code according to the parsed AST.

The regular expression parsing grammar is based on the following principles:

- it recognizes separately regular characters and metacharacters (except for ^ and $ which are intentionally indexed as part of the n-grams as explained above), because metacharacters must not be included into the n-grams search and hence represent a separator between n-grams,
- it recognizes character classes (enclosed in `[` and `]` brackets) which must be kept as a single token as they represent a single character position,
- it recognizes escaped sequences so that they can be handled correctly (possibly de-escaped),
- it recognizes repetitions which must be either entirely omitted (in case zero number of repetitions is allowed and hence the respective string portion

is entirely optional) or if at least one occurrence is obligatory, the operand character is duplicated and forms a suffix of previous n-grams and prefix of next n-gram. E.g. `abc+de` gets expanded into two n-grams `abc` and `cde` since every matching string must contain these.

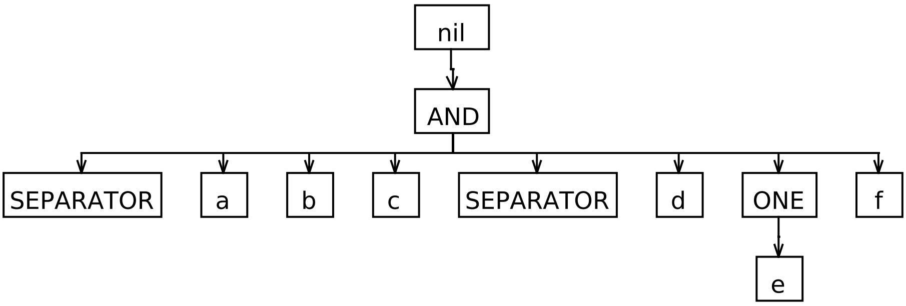


Fig. 2: Sample abstract syntax tree for the input query `.abc.*de+f`. Both `.` and `.*` substrings become a separator, while `e+` substring is recognized so as to search for `de` and `ef`.

Two sample AST's are provided in Figures 2 and 3. The tree walking parser looks up the found n-grams (in this sample `abc`, `de` and `ef` and combines the related source attribute IDs using the logical `AND` operator into a single stream of IDs that represent possibly matchings strings. Only these IDs are then subject to evaluation of the regular expression instead of the full lexicon. An overview of the whole dataflow is presented in Figure 4.

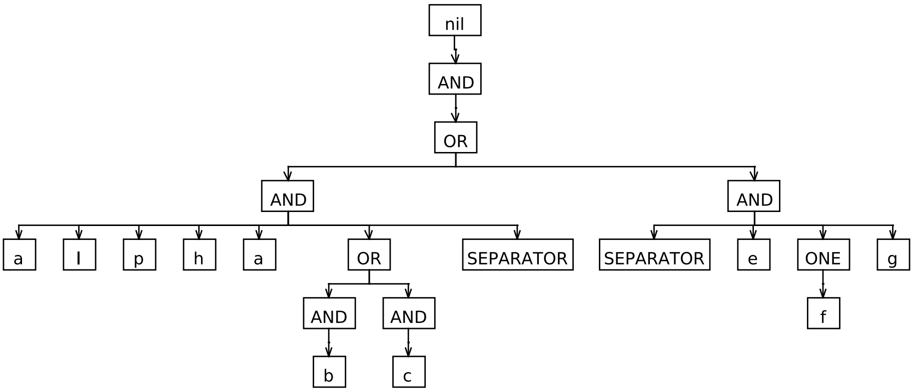


Fig. 3: Sample abstract syntax tree for the input query `(alpha(b|c).*|d*ef+g)`

The lexicon lookup for particular n-grams is either a direct mapping of an n-gram to ID using the dynamic attribute's lexicon, or in case of n-grams containing regular expression character class again an evaluation of a regular expression – however on a much smaller lexicon. In Table 1 we provide a comparison of original and n-gram lexicon sizes of various corpus and attribute combinations showing that the n-gram lexicon is usually by orders of magnitude smaller. It is obvious that for attributes with very small lexicons (such as tags based on atomic tagsets), the optimization is not worth doing and may even slow down the processing (as in the case of English and Japanese tags), therefore in the current implementation the n-gram indices are being compiled only for attributes with lexicons exceeding 10,000 items.

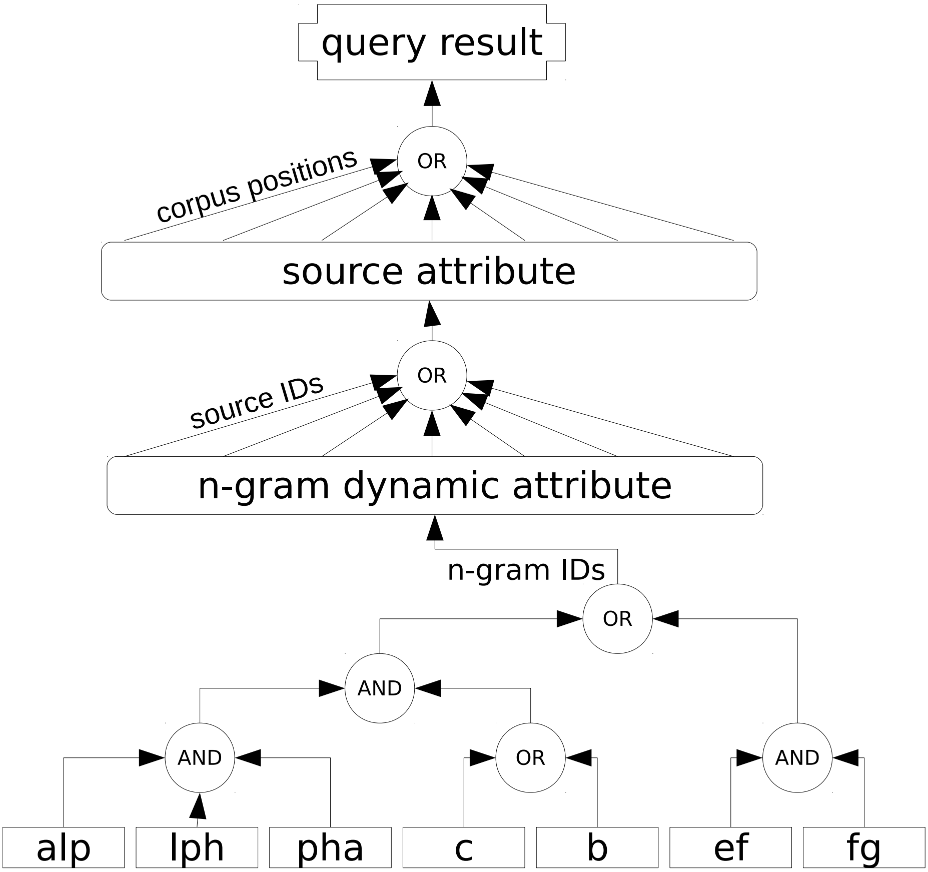


Fig. 4: Overview of the evaluation workflow using the n-gram prefetching optimization for input query `(alpha(b|c).*|d*ef+g)`

4 Evaluation

The optimization has been evaluated on a number of regular expressions executed against various corpora under the following conditions:

- a single evaluation thread running on an Intel(R) Xeon(R) CPU E5506 @2.13GHz,
- hot cache – the best time of three consecutive runs was counted,
- we have measured the time to retrieve first 20 hits – this includes the regular expression evaluation plus a number of other operations on the resulting positions stream but it is more representative from the user perspective (the processing within Sketch Engine is asynchronous and as soon as first 20 hits are available they are displayed to users),
- we provide the number of regular expression evaluations (calls to the PCRE matching function) with and without the optimization.

It follows from the evaluation that the speedup ranges from rather negligible 1.12 to enormous 100-times and the speedup ratio depends on a number of circumstances:

- obviously the larger the lexicon size of the source attribute, the more speedup can be achieved, and the evaluation shows that the additional indexing pays off only in cases where the lexicon size exceeds about 10,000 items
- for very small lexicons (e.g. in the case of the tag attribute in English corpora following the Penn Treebank tagset with only 60 atomic tags), the n-gram prefetching is not very beneficial as it enlarges the lexicon size,
- the n-gram prefetching is beneficial even in cases where the prefix optimization has already been previously in operation which is important since these two optimizations cannot be combined,[2]
- not surprisingly the n-gram prefetching is most beneficial for regular expressions containing rare n-grams (e.g. `strč` in Czech) but even for frequent n-grams (like `ten` in English) the speedup is usually around 20,
- even where the optimization itself involves regular expression evaluation (character class matching as in `[sz]p.*`), the speedup is significant and present even in comparison with prefix optimization (as in `pr[oe].*`).

5 Technical Notes

For creating the n-gram optimization index, there is a new tool included in Manatee called `mkregexattr` with a straightforward usage:

```
mkregexattr <CORPUS> <ATTRIBUTE>
```

[2] The prefix optimization results – by its nature – in a list of IDs sorted alphabetically, not numerically, and hence cannot be as such subject to any AND/OR stream operations.

Table 2: Evaluation of n-gram prefetching optimization. #RE denotes the number of regular expression matching function executions without and with n-gram prefetching, time is the evaluation time to acquire first 20 hits in seconds, S denotes the achieved speedup.

corpus	query	#RE w/o	#RE w/	time w/o	time w/	S
czTenTen12	`[word=".*ější"]`	18,978,703	41,426	10.030	0.341	29.4
	`[lemma=".*strč.*"]`	14,151,454	888	6.601	0.066	100.0
	`[tag="k1.*c4.*"]`	1,357	251	0.058	0.049	1.2
	`[word="[sz]p.*"]`	18,978,703	115,347	10.023	0.698	14.4
enTenTen12	`[word=".*ing"]`	27,894,538	913,004	21.931	2.768	7.9
	`[lemma=".*ten.*"]`	26,426,200	195,758	22.163	1.218	18.2
	`[word="pre.*ed"]`	80,054	7,553	0.294	0.178	1.7
	`[word="pr[oe].*"]`	251,924	198,297	1.329	0.920	1.4
	`[word=".*[dt]"]`	27,894,538	3,466,379	41.100	8.538	4.8
	`[tag="N.*"]`	60	5	0.056	0.048	1.2
jpTenTen11	`[word=".*ち.*"]`	13,844,200	30,160	8.182	0.364	22.4
	`[lemma=".*ア.*ス"]`	13,303,479	69,228	8.388	0.450	18.6
	`[word="ンテ.*"]`	17,078	17,077	0.199	0.178	1.12

The tool is part of Manatee version 2.111 and since this version, it is called automatically by `encodevert` at the end of corpus compilation for each attribute whose lexicon size exceeds 10,000 items.

6 Conclusions and Future Work

In this paper we have presented n-gram prefetching – an optimization approach for regular expression evaluation suitable for any corpus management system based on inverted indices and independent of any third-party regular expression library. We have shown that this optimization can significantly speedup user queries consisting of regular expressions. The idea has been practically implemented within the Manatee corpus management system used within the Sketch Engine corpus system and is part of the GPL-licensed part of Manatee available also within the open source NoSketch Engine suite[3].

In the future there might be a number of further optimizations that could be explored, starting with extending the character class recognition support to escape sequences like `\w`, `\d` etc., or dividing the n-gram lexicon into separate ones for uni-, bi-, and trigrams. A different kind of optimization may also lie in trying different regular expression library than PCRE or enabling PCRE just-in-time (JIT) features that are currently not in use – however such a contribution

[3] `http://nlp.fi.muni.cz/trac/noske`

is hard to evaluate as it very much depends on particular types of regular expressions and with regard to them the speedup might be fairly unstable.

Acknowledgements This work has been partly supported by the Ministry of Education of CR within the LINDAT-Clarin project LM2010013.

References

1. Rychlý, P.: Korpusové manažery a jejich efektivní implementace. PhD thesis, Fakulta informatiky, Masarykova univerzita, Brno (2000)
2. Evert, S., Hardie, A.: Twenty-first century corpus workbench: Updating a query architecture for the new millennium. (2011)
3. Knuth, D.E.: Retrieval on secondary keys. The art of computer programming: Sorting and Searching **3** (1997) 550–567
4. Rychlý, P.: Manatee/Bonito - A Modular Corpus Manager. In: Proceedings of Recent Advances in Slavonic Natural Language Processing 2007, Brno, Masaryk University (2007)
5. Hazel, P.: PCRE: Perl compatible regular expressions (2005)
6. Pomikálek, J., Rychlý, P., Jakubíček, M.: Building a 70 billion word corpus of English from ClueWeb. In: Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12). (2012) 502–506
7. Jakubíček, M., Rychlý, P., Kilgarriff, A., McCarthy, D.: Fast syntactic searching in very large corpora for many languages. In: PACLIC 24 Proceedings of the 24th Pacific Asia Conference on Language, Information and Computation, Tokyo (2010) 741–747
8. Jakubíček, M., Šmerk, P., Rychlý, P.: Fast construction of a word-number index for large data. In A. Horák, P.R., ed.: RASLAN 2013 Recent Advances in Slavonic Natural Language Processing, Brno, Tribun EU (2013) 63–67
9. Kilgarriff, A., Baisa, V., Bušta, J., Jakubíček, M., Kovář, V., Michelfeit, J., Rychlý, P., Suchomel, V.: The Sketch Engine: ten years on. Lexicography **1** (2014)
10. Parr, T.: The definitive ANTLR reference: building domain-specific languages. Pragmatic Bookshelf (2007)

Annex 1: ANTLR3 lexer and parser grammar for regular expressions

```
LPAREN:      ’(’;
RPAREN:      ’)’;
LBRACKET:    ’[’;
RBRACKET:    ’]’;
LBRACE:      ’{’;
RBRACE:      ’}’;
BINOR:       ’|’;
STAR:        ’*’;
PLUS:        ’+’;
QUEST:       ’?’;
DOT:         ’.’;
ZEROANDMORE: ’{0,}’ | ’{0,’ (’0’..’9’)+ ’}’;
ONEANDMORE:  ’{1,}’ | ’{1,’ (’0’..’9’)+ ’}’;
ESC:         ’\\’ (STAR | PLUS | QUEST | LBRACE | RBRACE | LBRACKET | RBRACKET
                  | LPAREN | RPAREN | DOT | BINOR | ’\\’ | ’^’ | ’$’
                  );
BACKREF:     ’\\’ (’0’..’9’)+;
SPECIAL:     ’\\’ .;
NOMETA:      ~(STAR | PLUS | QUEST | LBRACE | RBRACE | LBRACKET | RBRACKET
              | LPAREN | RPAREN | DOT | BINOR | ’\uFFFF’
              );
ENUM: LBRACKET CHARCLASS+ RBRACKET;
fragment CHARCLASS: ( (NOMETA|DOT) ’-’ (NOMETA|DOT)
           | ALNUM | ALPHA | BLANK | CNTRL | DIGIT | GRAPH | LOWER
           | PRINT | PUNCT | SPACE | UPPER | XDIGIT
           | (NOMETA|DOT)
          );


regex
    : regalt (BINOR^ regalt)* EOF!
    ;

regalt
    : regpart+ -> ^(AND regpart+)
    ;

regpart
    : regterm
      ( repet_one -> ^(ONE regterm)
      | repet_zero -> SEPARATOR
      | -> regterm
      )
    ;

regterm
    :
```

```
    (
        LPAREN regalt (
                        (BINOR regalt)+ -> ^(BINOR regalt regalt)
                        | -> regalt
                      ) RPAREN
    |   ENUM -> ENUM
    |   re_str -> re_str
    )
    ;

re_str
    : (DOT|BACKREF|SPECIAL) -> SEPARATOR
    | NOMETA -> NOMETA
    | ESC -> ESC
    ;

repet_one
    : PLUS -> PLUS
    | ONEANDMORE -> ONEANDMORE
    ;

repet_zero
    : QUEST -> QUEST
    | STAR -> STAR
    | ZEROANDMORE -> ZEROANDMORE
    ;
```

Modus Questions: Query Models and Frequency in Russian Text Corpora

Victoria V. Kazakovskaya[1] and Maria V. Khokhlova[2]

[1] Russian Academy of Sciences (Institute for Linguistic Studies),
Tuchkov per. 9, 199053, Saint-Petersburg, Russia
victory805@mail.ru
[2] Saint-Petersburg State University,
Universitetskaya nab. 11, 199034, Saint-Petersburg, Russia
khokhlova.marie@gmail.com

Abstract. The paper deals with the analysis of modus questions used in dialogues of native Russian speakers, discusses their quantitative properties and characteristics. The research focuses on the development of models describing these questions based on the Russian National Corpus and a newspaper corpus. The results obtained can be applied in various fields of natural language processing, e.g. dialogue systems.

Keywords: text corpora, dialogue, modus questions, Russian

1 Introduction

Modus questions (*MQs*) are interpreted as being question constructions appealing to the intentions of the addressee – their opinion, knowledge, evaluation or explanation. In this sense they are paramount for assimilation of the world of the subjective (the so-called theory of mind) [1]. Propositional attitudes and proposition of modus were studied in many works (see for example [2,3,4,5]). The scope of our research is limited to the analysis of models describing modus questions in Russian texts, particularly the quantitative properties of these types of questions in dialogues and the characteristics of the reactions (replies) given.

Prototypical MQs are represented by constructions containing explicit modus – modus frame[3]: e.g.

1. *Как ты думаешь (вы думаете) / полагаешь (вы полагаете) / считаешь (вы считаете)…?* 'What/How do you: SG or PL think / suppose etc. ?';
2. *Ты думаешь (вы думаете) / полагаешь (вы полагаете) / считаешь (вы считаете), что…?* 'Do you: SG or PL think / suppose etc. that…?';

[3] We follow Bally's differentiation between modus and dictum [6].

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2014, pp. 49–55, 2014. © Tribun EU 2014

3. *Почему (отчего) ты так думаешь (вы думаете) / полагаешь (вы полагаете) / считаешь (вы считаете)…?* 'Why do you: SG or PL think / suppose etc. that…?'.

Within dialogic units (or discourse sequences) MQs can occupy not only an initiative position (see above 1 – 3), but also a reactive one: e.g.

4. *– Да что же ты, не можешь стукнуть кулаком по столу? – **Почему не могу?*** 4a. *Ты думаешь, отчего у меня этот синяк под глазом?* (RNC) 'What are you, you cannot bang your fist on the table? **Why cannot I?** You think, why I have this black eye?'.

The size of modus, means of its expression and types of frame pattern vary extensively in utterances. For example, a completely verbalized modus can be the main predicative part of a compound sentence with an explanatory close (4a).

Incomplete (reduced) modus frames are often presented by parenthesis:

5. *Что, **по-вашему**, это может изменить <…>?* (RN) 'What, **in your opinion**, this can change <…>?'.

Being incorporated in the dictum, modus complicates a simple sentence with an analytic (namely, compound nominal) predicate:

6. *– Неужели вы его, правда, считаете величайшим? — спросил меня тот, кого мы в нашем рассказе условно называем Петровым. <…>* (RNC) 'Do you have him, however, considered the greatest? — Asked me the one whom we in our story conventionally call Petrov. <…>'

2 Materials and Methods

The research was carried out on corpora of the Russian language (compiled by S. Sharoff): 1) a subset of the Russian National Corpus (RNC, 116 million tokens) and 2) a Newspaper Corpus (RN, 70 million tokens) [7]. For the comparative analysis we used data from the Russian National Corpus [8].

Lexico-syntactic models [9] can be used for describing patterns involving lemmas or word forms, part-of-speech tags, characters, and other attributes in an annotated corpus. While writing lexico-syntactic models we used regular expressions and query language IMS Corpus Workbench.

The search of the system is based on morphological annotation combined with lemmata and word forms. For example, the pattern `[lemma="как"] []{0,5} [word="считаешь" | word="считаете"] []{0,15} [lemma="?"]` describes constructions with the interrogative-relative pronoun *как* ('how') and verb *считать* ('consider/think') in both 2SG or PL forms, with the distance between them being up to 5 words: cf. *как ты считаешь, …* 'how/what do you consider…', *как ты все-таки считаешь, …* 'how do you ever consider…', *как же ты все-таки считаешь, …* 'how do you ever still consider…'.

The restriction [ ]{0,15} means that there are up to 15 words between *считаешь (считаете)* 'you consider' and the end of the sentence (a question mark).

The above-discussed models underlying modus questions make up about 1.5% of the total number of question sentences (strictly speaking, words before a question mark) in the Russian National Corpus.

The whole range of modus questions was restricted to the most characteristic models – constructions that include modus frames of mental semantics with the prototypical intentional predicates (*полагать* 'suppose', *считать* 'consider', *думать* 'think') in the second person singular and plural forms inherent to the Russian replication.

3 Analysis Results

The constructions with mental predicates *считать* 'consider' (*считаешь / считаете?*) and *думать* 'think' (*думаешь / думаете?*) prove to be the most commonly used in the given corpora (see table 1).

In both corpora MQs with verbs in plural form are very frequent, prevailing in RN (about 90%). This fact could be due to the specifics of the corpus, i.e. a high number of interviews (in Russian the polite form is identical to the 2PL one).

The usage of the pronouns *ты* 'you': 2SG and *вы* 'you': 2PL being non-obligatory, nevertheless dominates in both corpora. In RNC there are 127 examples of constructions ***Вы (ты)*** *полагаете (полагаешь)…?* 'Do you: 2SG or 2PL suppose…?' and only 38 cases where the pronoun is omitted[4], while in RN there are 65 MQs with pronouns and only 5 sentences without them:

7. *<…>Полагаете, они вас услышали? — Думаю, что нет, к сожалению* (RN). '[Do you] suppose, they heard you? — I think not, unfortunately.'

The search within sentence boundaries can produce some difficulties because a punctuation mark is viewed as a token and has its own tag. Imposing restrictions for search within certain boundaries (e.g. <s> tags for a sentence) can be seen as a solution in this case.

The description of four modus models of question extracted from the corpus data are given below: *КАК-model* 'HOW-model', *НЕУЖЕЛИ-model* 'REALLY-model', *ПОЧЕМУ-model* 'WHY-model' and its version *ОТЧЕГО-model* 'WHY-model'.

[4] Russian language belongs to the so-called pro-drop languages.

Table 1: Modus questions in corpora

№	Модель	RN	RNC
1	как полагаешь / полагаете … ?	35	65
2	как думаешь / думаете … ?	445	939
3	как считаешь / считаете … ?	420	374
4	думаешь / думаете … ?	>987	>997
5	полагаешь / полагаете … ?	70	165
6	считаешь / считаете … ?	>993	>997
7	почему … полагаешь / полагаете … ?	3	6
8	почему … думаешь / думаете … ?	15	177
9	почему … считаешь / считаете … ?	36	63
10	неужели … полагаешь / полагаете … ?	0	9
11	неужели … думаешь / думаете … ?	9	132
12	неужели … считаешь / считаете … ?	2	19
13	отчего … полагаешь / полагаете … ?	0	1
14	отчего … думаешь / думаете … ?	0	3
15	отчего … считаешь / считаете … ?	0	0

3.1 КАК (считаешь / думаешь / полагаешь)-model

The pattern identifying the model in a corpus can be written in the following form: `[lemma="как"] []{0,5} [word="считаешь" | word="считаете"] []{0,15} [lemma="\?"]`.

An interrogative construction in the form of a composite sentence turns out to be the most typical for the КАК-model.

One can distinguish between several kinds of reply for YES /NO-questions based on a partial repetition of either the dictum part of the utterance or its modus part combined with relatives (*да* 'yes', *нет* 'no' etc.). Requests to agree or disagree with the previous information, to clarify it or to make an assumption can be found in the semantics of the reactions. See:

(a) repetition of the dictum (predicate):

8. – ***Как ты думаешь, нужен*** *будет нам еще Пегий пес или нет? – Нет, не нужен, – опять же совершенно уверенно отвечал Кириск* (RNC). 'Do you think we still need a Spotted Dog or not? – No, it is not needed – replied Kirisk, again, quite confidently.'

(b) repetition of the modus (also with negative particle *не* 'not'):

9. – ***Как вы думаете****, закон об ограничении пивной рекламы Госдума поддержит? — Думаю, да* (RNC). 'Do you think the law limiting beer advertising will be supported by the State Duma? – I think so.'

(c) repetition of parts of the modus and dictum:

10. – ***Как Вы считаете, поднимут*** *«Курск» в этом году или нет? – Считаю, что поднимут* (RN). 'Do you think [they] will raise the 'Kursk' this year or not? – I believe that [they] will raise it.'

(d) avoidance of an answer, clarifying or counter-questions:

11. – ***А вы, Таня, как считаете?*** – *Мне-то какое дело? – дернула Таня плечиком* (RNC). 'And you, Tanya, what do you think? – What do I care? – Tanya answered with a shrug of her shoulders.'

Replies to WH-questions are quite similar to the previous ones containing the requested information (dictum) that can be presented by a frame repeating a mental predicate:

12. – *Срок...* ***Как сам думаешь?*** *(= какой срок?) –* ***Думаю****, неделя (RNC)*. 'The term ... How long do you, yourself, think? (= How long?) – I think, a week.'

3.2 НЕУЖЕЛИ (считаешь / думаешь / полагаешь)-model

The present model can be seen as a modal complicated variant of the KAK-model and is more frequent in RNC compared to RN (160 vs. 11 examples). The pattern is as follows: `[lemma="неужели"] []{0,5} [word="считаешь" | word="считаете"] []{0,15} [lemma="\?"]`.

This question is represented by both complex and simple constructions (the latter is more typical for the predicates *полагать* and *считать*).

The replies include the following items:

(a) repetition of the dictum or its fragments:

13. – *Неужели думаешь, что игре удастся избежать политизации, реваншистского акцента? <...>. — Игре, может, и не удастся, а я в это все лезть не хочу* (RNC). 'Do you really think that the game will be able to avoid politicization, revanchist accent? <...> – The game maybe will not be able to avoid this, I do not want to interfere in it.'

(b) repetition of the modus:

14. – *Мама, неужели ты всерьез считаешь, что он мне пара? – Я ничего не считаю, я только вижу, что он тебя любит.* (RNC) 'Mom, do you seriously believe that we are a couple? —– I believe/consider nothing, I only see that he loves you.'

(c) repetition of parts of the modus and dictum:

15. — *Я здесь родился, я люблю этот город, неужели вы думаете, что я хочу оставить о себе плохую память? — Я просто думаю, что на вас давят интересы не столько эстетического плана, сколько денежного.* (RN) 'I was born here, I love this city, do you really think that I want to leave a bad memory about myself? — I just think that you are under financial influences rather than aesthetic ones.'

(d) avoidance of an answer, clarifying or counter-questions:

16. — *Неужели, как Хрущев, считаете, что они не умеют рисовать? — Хотите байку? Раз в Берлине сидим треплемся с товарищем — галерейщиком* <...>. (RN) 'Do you really think, like Khrushchev, that they do not know how to draw? — Would you like to hear a story? Once, in Berlin, we were sitting chatting with a friend who is a gallery owner <...>.'

3.3 ПОЧЕМУ (считаешь / думаешь / полагаешь)-model

The pattern is described as follows: `[lemma="почему"] []{0,5} [word="считаешь" | word="считаете"] []{0,15} [lemma="\?"]`.

The replies include the following items:

(a) repetition of the dictum (the requested information — the reasoning of the point of view) introduced by the subordinate conjunction потому что:

17. – *Почему вы так думаете? – Потому что очень хорошо его знаю! – с мстительным удовольствием сказала Марьяна* (RNC). 'Why do you think so? — Because I know him very well! — Mariana said, with vindictive pleasure. '

(b) repetition of the modus (mental predicates) that does not contain the reasoning of the opinion as the addressee disagrees:

18. *К. Почему вы так думаете? П. Тут и думать нечего.* <*Никто из наших деревенских Степшу убить не мог*> (RNC) 'K. Why do you think so? P. There is nothing to think about. No one from our village could (not) kill Stepsha.'

3.4 ОТЧЕГО (считаешь / думаешь / полагаешь)-model

The pattern is described as follows `[lemma="отчего"] []{0,5} [word="полагаешь" | word="полагаете"] []{0,15} [lemma="\?"]`.

This model is a variant of the ПОЧЕМУ-model being the least frequent in the corpora: there are 0 examples in RN, and there are only 4 examples in RNC.

Like the above-mentioned, the replies partially repeat the modus or exemplify the motivation of the point of view:

19. *– Но отчего вы так думаете? – спросил он. – Да хотя бы оттого, что в конце концов я возвращаюсь в реальный мир, — сказал я* (RNC). 'But why do you think so? – he asked. – If only because, in the end, I go back to the real world – I said.'

4 Conclusion and Further Work

The paper presents preliminary results of the study of interrogative constructions in Russian. The analysis shows that modus questions, although not being very frequent in dialogues of native Russian speakers nevertheless represent a certain trait of modern discourse. The *КАК считаете / думаете-model* proves to be the most prototypical one amongst the above-described models. Constructions appealing to the reasoning of the addressee's point of view as well as stylistically marked ones are less common.

References

1. Kazakovskaja, V.V.: Modusnyje voprosy (k probleme izbytochnosti). In: Acta Linguistica Petropolitana: Trudy Instituta lingvisticheskikh issledovanij RAN. Izbytochnost' v grammaticheskom stroje jazyka. Vol. 6, pp.225–242. Nauka, Sankt-Peterburg (2010)
2. Field, H.: Mental representation. In Block N. (ed) Readings in the Philosophy of Psychology 2, pp.78–114. CUP, Cambridge, Harvard University Press, MA (1978)
3. Gak, V.G.: O logicheskom ischislenii semanticheskikh tipov propozitsional'nykh glagolov. In Propozitsional'nyje predikaty v logicheskom i lingvisticheskom aspekte. pp.38–70. Nauka, Moskva (1987)
4. Jaszczolt, K.M. (ed): The pragmatics of propositional attitude reports. Elsevier Science, Oxford (2000)
5. Larson, R.: The grammar of intentionality. In Peter G., Prayer G. (eds) Logical form and language, pp.228–262. Clarendon press, Oxford (2002)
6. Bally, Ch.: Linguistique générale et linguistique française. Leroux, Paris (1932)
7. Corpora of the Russian language, `http://corpus1.leeds.ac.uk/ruscorpora.html`
8. Russian National Corpus, `http://ruscorpora.ru`
9. Khokhlova, M.V.: Razrabotka grammaticheskogo modulja russkogo jazyka dlja spetsial'noj sistemy obrabotki korpusnykh dannykh. In: Vestnik Sankt-Peterburgskogo gosudarstvennogo universiteta. Ser. 9. Filologija, vostokovedenije, zhurnalistika. Vol. 2, pp.162–169. Izdatel'stvo Sankt-Peterburgskogo gosudarstvennogo universiteta, Sankt-Peterburg (2010)

Low Inter-Annotator Agreement
=
An Ill-Defined Problem?

Vojtěch Kovář, Pavel Rychlý, and Miloš Jakubíček

Faculty of Informatics
Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
{xkovar3,pary,jak}@fi.muni.cz

Abstract. Annotation tasks where the inter-annotator agreement is low are usually considered ill-defined and not worth attention. Such tasks are also considered unsuitable for algorithmic solution and for evaluation of computer programs that aim at solving them. However, there is a lot of problems (not only) in the natural language processing field that are practically defined and do have this nature, and we need computer programs that are able to solve them.
The paper illustrates such problems on particular examples and suggests methodology that will enable training and evaluating tools using data with low inter-annotator agreement.

Keywords: NLP, inter-annotator agreement, low inter-annotator agreement, evaluation

1 Introduction

Inter-annotator agreement (IAA) is considered one of the key indicators of whether a particular classification task is well-defined or not. A lot of attention has been paid to the IAA problem [1,2,3] aiming at not only measuring the agreement, but also excluding the expected amount of agreements by chance, and interpretation of different values of the IAA measurements.

The task is generally considered well-defined, if the inter-annotator agreement is very high, and ill-defined and not worth attention, if the agreement is rather low. To our best knowledge, this is a common view for all classification tasks, through all scientific fields.

In the field of natural language processing (NLP), however, there is a huge number of tasks that do not have naturally high inter-annotator agreement, as people do not agree on the right annotation (even if they are well-educated specialists). Even for such a seemingly straightforward task as morphological tagging (of English), the reported IAA is around 97 percent [4], for more complex tasks like syntactic analysis, information extraction or question answering, it is much much less (e.g. [5]).

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2014, pp. 57–62, 2014. © Tribun EU 2014

This discrepancy (need for high IAA vs. naturally low IAA in case of most of the NLP tasks) leads to undesirable side-effects. On one side, there are extremely extensive manuals for annotation [6,7] containing hundreds of pages. On the other hand, inter-annotator agreement is rarely published – e.g. the reported IAA for English morphological tagging comes from a semi-official note [4], for Prague Dependency Treebank (PDT [8]), the primary syntactic resource for Czech, there is only one report that describes the annotation of a specific sub-part of PDT [9] which does not report very high numbers in case of important parts of the annotation. It is of course just a speculation, but our opinion is that the results are not published because low IAA numbers would put the whole (mostly very costly) resources in a bad light.

We think both of these effects are really bad, as the aim of all NLP tasks is to learn computers what humans are able to do without any manuals – understand the language – so there should be only minimalistic instructions for any NLP annotation. On the other hand, ambiguity (and low agreement rate) is natural – people often read same sentences differently, and often have to ensure that they understand each other correctly. We can say that low IAA is an integral property of natural languages.

Therefore, we need to be able to handle the tasks with low IAA and use the data with low IAA in training and evaluations, rather than ignore them or try to overcome the fact that the language is ambiguous. This paper suggests a method for using the data with low IAA for meaningful evaluations. We discuss the requirements on such a method, and also some drawbacks and limits of the proposed approach.

2 Problem Examples

In this section we provide real-world examples of the tasks where low IAA causes problems.

2.1 Syntactic Annotation

An example of the project that tried to solve low IAA by extensive manuals for annotation, is syntactic annotation in the Prague Dependency Treebank (PDT [8]), a leading syntactic resource for Czech that we have already mentioned. The manual on the analytical (syntactic) layer has about 300 pages [7], and the annotation procedure was as follows.

Each sentence was annotated by 2 independent annotators, and where they did not agree, there was a third (more experienced) annotator to judge them [8]. So, the one and only syntactic representation available in the treebank is often based on 2/3 biased agreements according to a very complex manual. This procedure is error-prone, and also many of the rules in the manual are debatable. Some of the resulting problems are discussed in [10].

But mainly, the procedure goes against the ambiguous character of the language: In sentences like *"A plane crashed into the field behind the forest"*, it does not matter for correct understanding the sentence, whether the phrase *"behind the forest"* depends on *"crashed"* or *"field"* (although in similar sentences it may be very important). The resulting information is the same. But the annotators need to decide it, and so do the syntactic parsers that are trained and evaluated using this data. This does not correspond to the analysis procedure as it happens when humans analyze the text.

And this is just one example of frequent syntactic ambiguity of many.

2.2 Collocation and Terminology Extraction

Extraction of collocations is an important task for language learners and dictionary makers, to learn or record that in English one says "strong tea" rather than "powerful tea" or "light a fire" instead of "make a fire". However, the agreement among lexicographers on what is good collocation and what is not, is very low [11].

On the other hand, the automatic applications for collocation extraction (e.g. Sketch Engine [12]) are present and they are commercially interesting. They just did not undergo a proper scientific evaluation yet (the procedure reported in [11] is rather debatable, as it uses the same methodology that is used in classification tasks with high IAA), as there is no methodology for evaluating tasks with such a large grey (disagreement) zone.

For extraction of terminology from domain-specific texts, there is nearly the same situation. Terminology (e.g. in form of list of terms) is needed for terminology dictionaries, language learners and consistent translations, and the applications are already there (e.g. [13]). But the agreement on what is and what is not a term in a given domain is very low, and proper evaluation is missing, as there is no methodology available.

The three examples above only illustrate the problem – there are many similar tasks that are neither solved nor evaluated, as they are not "well-defined", however, they are needed and we need a methodology to evaluate them.

3 Methodology Proposal

In this section we present our proposal for annotation and evaluation of tasks with low IAA.

3.1 Requirements

We will illustrate requirements on a binary classification task, which is a typical case (most of the tasks can be straightforwardly reduced to a binary classification task). Let us have classes *Positive* and *Negative*, and the task is to assign each data item to one of these classes. We want to evaluate an automatic tool that does this classification in some imperfect way.

Then, let us have some "gold standard" data annotated by multiple human annotators, that agree in some cases, and disagree in others.

The automatic tool should get positive points for every item assignment into class *Positive*, where all the human annotators agreed that it should be class *Positive*. Similarly for *Negative*. The tool should get negative points for every item assignment into class *Positive*, where all the human annotators agreed that it should be class *Negative*, and vice versa.

We need to be able to handle cases where the annotators do not agree with each other. We propose taking these cases off the evaluation and not count them in at all. Because if even one of the annotators interprets the data differently, the general "human interpretation" is not clear and the automatic tool should get neither positive, nor negative points for any assignment, in these cases.

3.2 Proposed Procedure

Based on the requirements, we introduce a modification of the standard measures *precision* and *recall*, defined for unambiguous gold standard data without human disagreements as follows:

$$precision = \frac{\#true\_positives}{\#true\_positives + \#false\_positives}$$

$$recall = \frac{\#true\_positives}{\#true\_positives + \#false\_negatives}$$

The modified precision and recall will use the same formulas, but with different meaning of *true_positive*, *false_positive* and *false_negative*:

- *#true_positives* will be defined as number of data items where all the human annotations were *Positive* and our tool said *Positive*.
- *#false_positives* will be defined as number of data items where all the annotations were *Negative* but the output of our tool said *Positive*.
- *#false_negatives* will be defined as number of data items where all the annotations were *Positive* and our tool said *Negative*.

In other words, we firstly remove all the data items where the human annotators disagree, and then measure standard precision and recall on the rest.

This idea can be easily generalized to classification into more classes. In that case, however, we may want to give some positive points in addition if the output of our tool agreed *at least with one of the annotators*, and/or negative points if the output of our tool agreed *with none of the annotators, even if they did not agree*. In this case, we will be able to somehow use the data even if the annotators disagree. However, this approach brings more complexity into the evaluation and it may me better to transform the problem into a binary classification task, which is possible in most cases, and transparent.

4 Discussion

There are some difficulties with the above introduced procedure. In the following points we mention them and discuss possible solutions:

- On the first sight, the procedure increases the price of the testing data, as many of the annotations (all cases where the annotators disagreed) are not used. However, the data will record ambiguity and will be of higher quality than if we attempt to *decide* the disagreements. Therefore, also the evaluations will be more sound, and automated learning from such data will be able to be more informed.
- We need to count with random agreements, especially when the part of the data where people disagree is rather big. Probability of random agreements can be easily computed for most of the classification tasks, and can be trivially decreased by increasing the number of annotators. For example, if probability of *Positive* judgement is 50%, increasing number of annotators to 7 will reduce the number of random agreements below 1%. Of course, sometimes it will mean increasing costs again. On the other hand, in most of the tasks the probability of the *Positive* judgement is much lower.
- In cases where the agreement is really low, the question whether the task is well-defined or not, will persist. The maintainers of the data should check carefully if the level of disagreement corresponds to the real ambiguity of the task and correct the annotation instructions if not. It is not easy to introduce a quantitative algorithmic test here, as the level of ambiguity significantly varies among various tasks.

5 Conclusions

We have introduced a new view on classification problems where people often disagree, mainly from the perspective of natural language processing, but suitable for any other field. We have proposed a methodology for using data with disagreements for testing (and partly training) of automatic classification tools. The proposed method is straightforward and easily applicable to any data.

We believe that in the future the data with disagreements will not be considered radioactive and they will be used for serious research. Also, we believe that the idea will encourage data maintainers to publish their agreement figures consistently.

Acknowledgements This work has been partly supported by the Ministry of Education of the Czech Republic within the LINDAT-Clarin project LM2010013 and by the Czech-Norwegian Research Programme within the HaBiT Project 7F14047.

References

1. Cohen, J.: A coefficient of agreement for nominal scales. Educational and psychological measurement **20**(1) (1960) 37–46
2. Carletta, J.: Assessing agreement on classification tasks: The kappa statistic. Computational linguistics **22**(2) (1996) 249–254
3. Fleiss, J.L., Levin, B., Paik, M.C.: Statistical methods for rates and proportions. John Wiley & Sons (2013)
4. Manning, C.D.: Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? In: Computational Linguistics and Intelligent Text Processing - 12th International Conference, CICLing 2011. Springer, Berlin (2011) 171–189
5. Sampson, G., Babarczy, A.: Definitional and human constraints on structural annotation of English. Natural Language Engineering **14**(4) (2008) 471–494
6. Bies, A., Ferguson, M., Katz, K., MacIntyre, R., Tredinnick, V., Kim, G., Marcinkiewicz, M.A., Schasberger, B.: Bracketing guidelines for treebank II style Penn treebank project (1995) `languagelog.ldc.upenn.edu/myl/PennTreebank1995.pdf`.
7. Hajič, J., Panevová, J., Buráňová, E., Urešová, Z., Bémová, A., Štěpánek, J., Pajas, P., Kárník, J.: Annotations at analytical level: Instructions for annotators (2005) `ufal.mff.cuni.cz/pdt2.0/doc/manuals/en/a-layer/pdf/a-man-en.pdf`.
8. Hajič, J.: Complex corpus annotation: The Prague dependency treebank. Insight into the Slovak and Czech Corpus Linguistics (2006) 54
9. Mikulová, M., Štěpánek, J.: Annotation procedure in building the Prague Czech-English dependency treebank. In: Slovko 2009, NLP, Corpus Linguistics, Corpus Based Grammar Research, Bratislava, Slovakia, Slovenská akadémia vied (2009) 241–248
10. Kovář, V., Jakubíček, M.: Prague dependency treebank annotation errors: A preliminary analysis. In: Proceedings of Third Workshop on Recent Advances in Slavonic Natural Language Processing, Brno, Masaryk University (2009) 101–108
11. Kilgarriff, A., Rychlý, P., Jakubíček, M., Kovář, V., Baisa, V., Kocincová, L.: Extrinsic corpus evaluation with a collocation dictionary task. In N.C.C., Choukri, K., Declerck, T., Loftsson, H., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., Piperidis, S., eds.: Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14), Reykjavik, Iceland, European Language Resources Association (ELRA) (2014) 1–8
12. Kilgarriff, A., Baisa, V., Bušta, J., Jakubíček, M., Kovář, V., Michelfeit, J., Rychlý, P., Suchomel, V.: The Sketch Engine: Ten years on. Lexicography **1** (2014)
13. Kilgarriff, A., Jakubíček, M., Kovář, V., Rychlý, P., Suchomel, V.: Finding terms in corpora for many languages with the Sketch Engine. In: Proceedings of the Demonstrations at the 14th Conferencethe European Chapter of the Association for Computational Linguistics, Gothenburg, Sweden, The Association for Computational Linguistics (2014) 53–56

SkELL: Web Interface for English Language Learning

Vít Baisa and Vít Suchomel

Natural Language Processing Centre
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
{xbaisa,xsuchom2}@fi.muni.cz

Lexical Computing Ltd.
Brighton, United Kingdom
vit.{baisa,suchomel}@sketchengine.co.uk

Abstract. We present a new web interface for English language learning: SkELL. The name stands for Sketch Engine for Language Learning and is aimed at students and teachers of English language. We describe SkELL features and the processing of corpus data which is fundamental for SkELL: spam free, high quality texts from various domains including diverse text types covering majority of English language phenomena.

Keywords: Sketch Engine, concordance, thesaurus, word sketch, language learning, English language, corpus

1 Introduction

There are many websites for language learners: wordreference.com[1] and *Using English*[2] are just two of many. Some of them are using corpus tools or corpus data such as Linguee[3], Wordnik[4], bab.la[5]. They usually provide dictionary-like features: definitions and translation equivalents in selected languages. Some of them provide even examples from parallel corpora (Linguee).

We introduce here a new web interface aimed at teachers and students of English language which offers similar functions as above-mentioned tools but at the same time it is based on a specially processed corpus data suitable for the language learning purpose.

We call it SkELL: Sketch Engine for Language Learning. The Sketch Engine[6] is a state-of-the-art web-based tool for building, managing and exploring large

[1] http://www.wordreference.com/

[2] http://www.usingenglish.com/

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

[4] http://www.wordnik.com/

[5] http://en.bab.la/

[6] http://www.sketchengine.co.uk

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2014, pp. 63–70, 2014. © Tribun EU 2014

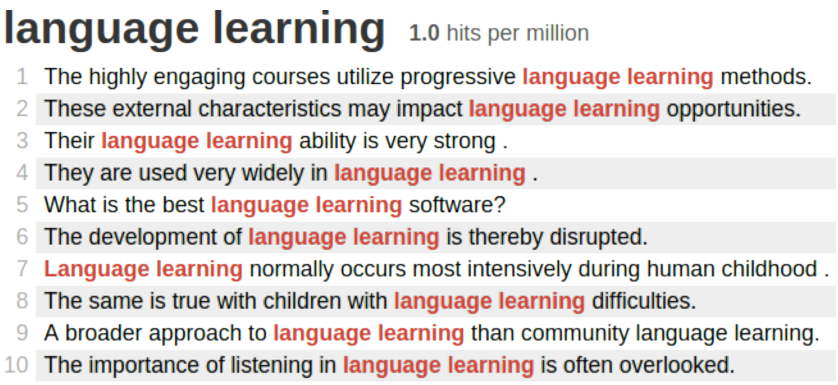
text collections in dozens of languages. SkELL is derived from the Sketch Engine and the data SkELL relies upon (SkELL corpus) is built using the very same tools as Sketch Engine uses: web crawler SpiderLing [1], tokeniser unitok.py [2] and TreeTagger [3]. Also, it uses a technique for scoring sentences according their appropriateness for using as example sentences in learners' dictionaries, GDEX [4].

2 Features of SkELL

SkELL features offer three ways for exploring the SkELL corpus. The first is the *concordance*: for a given word or phrase, it will return up to 40 example sentences. The second is the *word sketch* through which typical collocates for a given word can be discovered. And the third is *similar words* (thesaurus) which lists words that are similar to, though not necessarily synonymous with, a search word. The similar words are visualized with a word cloud. The web interface is optimized for mobile and touch devices.

SkELL features are built upon Bonito corpus manager [5] features. Bonito provides many standard functions as many other corpus managers: concordancing, word list generating, context statistics and also some advanced features like distributional thesaurus [6] and word sketches [7]. We have chosen these three: 1) concordance, 2) word sketch and 3) thesaurus (similar words).

2.1 Concordance (or examples)



language learning 1.0 hits per million

1 The highly engaging courses utilize progressive **language learning** methods.
2 These external characteristics may impact **language learning** opportunities.
3 Their **language learning** ability is very strong .
4 They are used very widely in **language learning** .
5 What is the best **language learning** software?
6 The development of **language learning** is thereby disrupted.
7 **Language learning** normally occurs most intensively during human childhood .
8 The same is true with children with **language learning** difficulties.
9 A broader approach to **language learning** than community language learning.
10 The importance of listening in **language learning** is often overlooked.

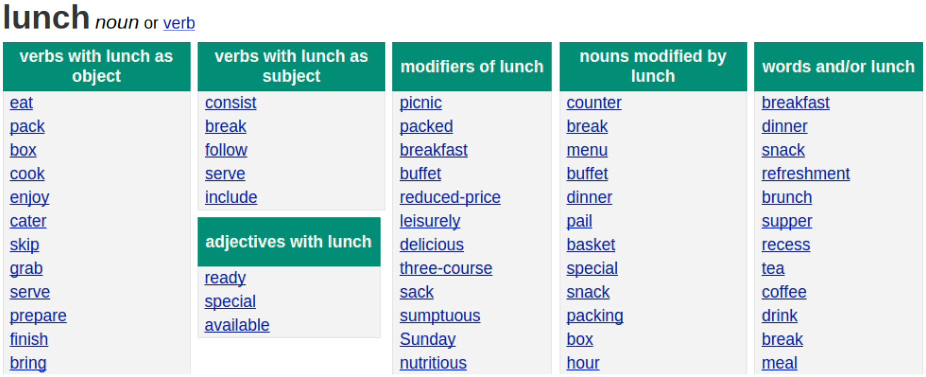
Fig. 1: Example of concordance for *language learning* phrase

Concordance offers a powerful full-text search tool. For a word or a phrase it returns up to 40 example sentences featuring the query words. Concordance feature is useful for discovering how words behave in English.

The search is case insensitive, i.e. it will yield the same results for *rutherford* and *Rutherford*. Moreover the results may contain the query (a word or a phrase) in a derived word form. For *mouse* (lemma) it will find sentences also with *mice*. For *mice* the result will contain a different set of sentences: only *mice* occurrences.

It is not necessary for users to specify part of speech (PoS, e.g. noun, verb, adjective, preposition, adverb etc.) of the search term is not necessary. If you search for *book*, it will give sentences with *book* as a verb and as a noun and both in various word forms (*booking*, *booked*, *books*).

2.2 Word sketch (or word profile)



lunch *noun* or verb

verbs with lunch as object	verbs with lunch as subject	modifiers of lunch	nouns modified by lunch	words and/or lunch
eat	consist	picnic	counter	breakfast
pack	break	packed	break	dinner
box	follow	breakfast	menu	snack
cook	serve	buffet	buffet	refreshment
enjoy	include	reduced-price	dinner	brunch
cater		leisurely	pail	supper
skip	**adjectives with lunch**	delicious	basket	recess
grab	ready	three-course	special	tea
serve	special	sack	snack	coffee
prepare	available	sumptuous	packing	drink
finish		Sunday	box	break
bring		nutritious	hour	meal

Fig. 2: Example of word sketch for *lunch*.

Word sketch function is very useful for discovering collocates and for studying contextual behaviour of words. Collocates of a word are words which occur frequently together with the word—they "co-locate" with the word. See [7] for more info.

For query *mouse*, SkELL will generate several tables containing collocates of the headword *mouse*. Table headers describe what kind of collocates (always in basic word form) they contain.

By clicking on a collocate, a concordance with highlighted headwords and collocates is shown. This way it can be seen how the two collocates together are usually used in English language.

By default, the most frequent PoS is shown in Word Sketches. If a word (*book*, *fast*, *key*, . . .) can have more than one PoS, the alternative links are shown next to the headword (see in Figure 2).

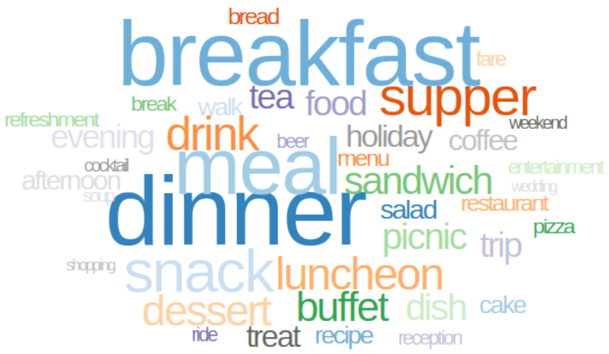


Fig. 3: Example of similar words in word cloud for *lunch*.

2.3 Similar words (or thesaurus)

The third functions serves for finding words which are similar (not only synonyms) to a search word. For a word (multi word queries are not supported yet) SkELL will return list of up to 40 most similar words visualized using wordcloud. The wordcloud is generated using D3.js library[7] and a wordcloud plugin[8].

As in Word Sketch if a word can have more than one PoS, the links to alternative results are provided.

3 SkELL corpus

SkELL is using a large text collection—SkELL corpus—gathered specially for the purpose of English language learning. It consists of texts from news, academic papers, Wikipedia articles, open-source (non)-fiction books, webpages, discussion forums, blogs etc. There are more than 60 million sentences in SkELL corpus and more than one billion words in total. This amount of textual data provides a sufficient coverage of everyday, standard, formal and professional English language.

In the following subsections we describe the most important data resources which have been used in building the SkELL corpus and the processing of the data.

3.1 English Wikipedia

One of the largest parts of SkELL corpus is English Wikipedia.[9] We have used Wikipedia XML dump[10] from October 2014. The XML file has been converted

[7] `http://d3js.org/`

[8] `http://www.jasondavies.com/wordcloud/about/`

[9] `https://en.wikipedia.org`

[10] `https://dumps.wikimedia.org/`

to plain text preserving only a few structure tags (documents, headings, paragraphs). using slightly modified script `WikiExtractor.py`[11]. We have filtered thousands of articles which are supposed to not contain fluent English text, e.g. articles named "List of ..." and then sorted all articles according their length and took top 130,000 articles. Among the longest articles there were e.g.: *South African labour law*, *History of Austria*, *Blockade of Germany*, ... It is clear that there are many articles from geographical and historical domains.

3.2 Project Gutenberg

The Project Gutenberg[12] (PG) focuses on gathering public domain texts in many languages. The majority of texts is in English. We have downloaded all English texts using *wget*[13]. and converted the HTML files to plain text.

The largest texts in English PG collection are *The Memoires of Casanova*, *The Bible (Douay-Rheims version)*, *The King James Bible*, *Maupassant's Original short stories*, *Encyclopaedia Britannica*, etc.

3.3 English web corpus, enTenTen14

We have prepared two subsets from the enTenTen14 [8] which has been crawled in 2014. The *White* (bigger) part contains only documents from web domains in `dmoz.org` or in the whitelist of `urlblacklist.com`. The *Superwhite* (smaller) containing documents domains listed in the whitelist of `urlblacklist.com` – a subset of *White* (in case there is still some spam in the larger part taken from `dmoz.org`)

Categories from the following list are allowed categories from `dmoz.org` in *Superwhite* part: 1) blog: journal/diary websites, 2) childcare: sites to do with childcare, 3) culinary: sites about cooking, 4) entertainment: sites that promote movies, books, magazine, humor, 5) games: game related sites, 6) gardening: gardening sites, 7) government: military and schools etc., 8) homerepair: sites about home repair, 9) hygiene: sites about hygiene and other personal grooming related stuff, 10) medical: medical websites, 11) news: news sites, 12) pets: pet sites, 13) radio: non-news related radio and television, 14) religion: sites promoting religion, 15) sportnews: sport news sites, 16) sports: all sport sites, 17) vacation: sites about going on holiday, 18) weather: weather news sites and weather related and 19) whitelist: sites specifically 100% suitable for kids. Finally we have decided to include the whole *White* part. It contained 1.6 billion tokens.

[11] `http://medialab.di.unipi.it/wiki/Wikipedia_Extractor`

[12] `http://www.gutenberg.org/`

[13] `http://www.gutenberg.org/wiki/Gutenberg:Information_About_Robot_Access_to_our_Pages`

3.4 WebBootCat corpus

One part of the SkELL corpus has been built using WebBootCat [9]. This approach uses seed words to prepare queries for a search engine. The pages from the search results are downloaded, cleaned and converted to plain text preserving basic structure tags. We assume the search results from the search engine are spam-free. We have run the tool several times with general English words as seed words yielding approximately 100 million tokens.

3.5 Other resources

The whole British National Corpus [10] has been also included. The rest of the SkELL corpus consists of free news sources. The Table 1 contains all the sources used in SkELL corpus.

Table 1: Sources used in SkELL corpus

Subcorpus	tokens	used
Wiki	1.6 G	500 M
Gutenberg	530 M	200 M
White	1.6 G	500 M
WebBootCated	105 M	all
BNC	112 M	all
other sources	344 M	200 M

3.6 Processing the data

When all the subparts have been gathered and pre-cleaned (we have removed all structures except sentences), we have run it through our standard processing pipe:

1. normalization: quotes, interpunction normalization,
2. tokenization: we have used `unitok.py`,[14]
3. TreeTagger[15] for English,
4. deduplication on sentence level: `onion` tool[16] has been used.

The corpus was then compiled using manatee indexing library [5]. Then we have scored all sentences in the corpus using GDEX tool [4] for finding good dictionary examples (sentences) for query results. All sentences in the corpus were sorted according to the score and saved in this order. This is a crucial part of the processing as it speeds up further querying of the corpus. Instead of

[14] `https://corpus.tools`

[15] `http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/`

[16] `http://nlp.fi.muni.cz/projects/onion/`

sorting good dictionary examples on-the-fly (which is used in Sketch Engine), all query results for concordance searches are presorted in the source vertical file.

The standard GDEX definition file used for English has been only slightly changed to prefer sentences with more frequent words, filtering out effectively all sentences with special terminology, typos and rare words (rare names). By default, short sentences are preferred, sentences containing inappropriate or spam words are scored lower.

The word sketch grammar required for computing word sketches has been modified: it contains only a few grammar rules with self-explanatory names.

3.7 Versioning and referencing

Since SkELL corpus may be changed in the future (further cleaned, refined, updated), all references to particular results of SkELL should be accompanied by the current version. The web interface may also be changed occasionally. That is why at the bottom of SkELL page, there is a version in this format: version1-version2. The first corresponds to a version of the web interface and the second to a version of SkELL corpus.

4 Conclusions and future work

The web interface is available at `http://skell.sketchengine.co.uk`. The version for mobile devices which is optimized for smaller screens and for touch interfaces is available at `http://skellm.sketchengine.co.uk`. If you access the former link from a mobile device it should be detected and redirected to the mobile version automatically.

We have described a new tool which we belive will turn out to be very useful for both teachers and students of English. The processing chain is ready to be used also for other languages. The interface is also directly reusable for other languages, the only prerequisite is the preparation of the corpus.

We are gathering feedback from various users and will refine the corpus data according it. In the future we plan these updates to SkELL:

1. to create a special grammar relation for English phrasal verbs,
2. to combine examples for collocations from word sketch collocates to build a more representative concordance for a given word,
3. to update the corpus with the newest text resources to provide examples for the newest trending words and neologisms,
4. to analyse access logs of SkELL and put favourite searches to the main page,
5. to provide commonest string [11] for word sketch collocates,
6. to allow multi word sketches which have been already introduced in [11],
7. to implement necessary methods for multi word thesaurus and
8. to build SkELL also for other languages (Russian, Czech, German and other).

Acknowledgement This work was partially supported by the Ministry of Education of CR within the LINDAT-Clarin project LM2010013 and by the Czech-Norwegian Research Programme within the HaBiT Project 7F14047.

References

1. Suchomel, V., Pomikálek, J., et al.: Efficient web crawling for large text corpora. In: Proceedings of the seventh Web as Corpus Workshop (WAC7). (2012) 39–43
2. Michelfeit, J., Pomikálek, J., Suchomel, V.: Text Tokenisation Using unitok. In Horák, A., Rychlý, P., eds.: 8th Workshop on Recent Advances in Slavonic Natural Language Processing, Brno, Tribun EU (2014) 103–109
3. Schmid, H.: Probabilistic part-of-speech tagging using decision trees. In: Proceedings of the international conference on new methods in language processing. Volume 12., Manchester, UK (1994) 44–49
4. Kilgarriff, A., Husák, M., McAdam, K., Rundell, M., Rychlý, P.: Gdex: Automatically finding good dictionary examples in a corpus. In: Proceedings of EURALEX. Volume 8. (2008)
5. Rychlý, P.: Manatee/bonito–a modular corpus manager. In: 1st Workshop on Recent Advances in Slavonic Natural Language Processing, within MU: Faculty of Informatics Further information (2007) 65–70
6. Rychlý, P., Kilgarriff, A.: An efficient algorithm for building a distributional thesaurus (and other sketch engine developments). In: Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions, Association for Computational Linguistics (2007) 41–44
7. Kilgarriff, A., Rychlý, P., Smrz, P., Tugwell, D.: The Sketch Engine. Information Technology **105** (2004)
8. Jakubíček, M., Kilgarriff, A., Kovář, V., Rychlý, P., Suchomel, V., et al.: The tenten corpus family. In: Proc. Int. Conf. on Corpus Linguistics. (2013)
9. Baroni, M., Kilgarriff, A., Pomikálek, J., Rychlý, P., et al.: Webbootcat: instant domain-specific corpora to support human translators. In: Proceedings of EAMT. (2006)
10. Leech, G.: 100 million words of english: the british national corpus (bnc). Language Research **28**(1) (1992) 1–13
11. Kilgarriff, A., Rychly, P., Kovář, V., Baisa, V.: Finding multiwords of more than two words. Proceedings of EURALEX2012 (2012)

Text Tokenisation Using `unitok`

Jan Michelfeit, Jan Pomikálek, and Vít Suchomel

Natural Language Processing Centre
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
{xmichelf,xsuchom2}@fi.muni.cz

Lexical Computing Ltd.
Brighton, United Kingdom
name.surname@sketchengine.co.uk

Abstract. This paper presents `unitok`, a tool for tokenisation of text in many languages. Although a simple idea – exploiting spaces in the text to separate tokens – works well most of the time, the rest of observed cases is quite complicated, language dependent and requires a special treatment. The paper covers the overall design of `unitok` as well as the way the tool deals with some language or web data specific tokenisation cases. The rule what to consider a token is briefly described. The tool is compared to two other tokenisers in terms of output token count and tokenising speed. `unitok` is publicly available under the GPL licence at `http://corpus.tools`.

Keywords: tokenisation, corpus tool

1 Introduction

Tokenisation of a text is the process of splitting the text to units suitable for further computational processing. It is an important data preparation step allowing to perform more advanced tasks like morphological tagging or parsing. Applying a good tokenisation method is necessary for building usable text corpora.

Using spaces in the text as token delimiters is a simple and quick way to tokenise text. In fact, the presented approach is based on this observation. However, there are many complicated cases to deal with which cannot be solved just by splitting tokens by space characters. Some cases are language dependent and require a special treatment. Other sequences to recognize as single or multiple tokens come from the web pages – a rich yielding source of data in text corpora [4].

The aim of this work was to develop a tokeniser

- fast – able to process big data in billion word sized corpora,
- reliable – robust to deal with messy web data,

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2014, pp. 71–75, 2014. © Tribun EU 2014

- universal – allowing at least basic support for all writing systems utilizing a whitespace to separate words[1],
- easy to maintain – adding new tokenisation rules or making corrections based on evaluation should be straightforward,
- text stream operating – text in, tokenised text (one token per line) out[2],
- reversible – the tokenised output must contain all information needed for reconstructing the original text[3]

The resulting tool called unitok has been developed since 2009. It has become a part of corpus processing pipeline used for many corpora [3,1,6] since then.

2 The Problem of Words

There are legitimate questions and related problems concerning the desired behaviour of a good tokeniser: What is a word, what is a sentence? [2] Our approach to unitok is based on the point of view of a corpus user. It is important to know what tokens the users search for in the corpus concordancer (and other corpus inspection tools) and what tokens they expect to figure in the corpus based analysis (such as word frequency lists, collocations (Word Sketches), thesaurus, etc.).

The answer is the users search for sequences of letters. Sequences of numbers, marks, punctuation, symbols, separators and other characters should be clustered together in order to be counted as single tokens in corpus statistics. The definition of the charater classes and mapping of each letter to a class has been made by The Unicode Consortium.[4]

3 Implementation

3.1 Related Work

A set of rules in flex[5] has been used by [5] to implement a tokeniser. We chose to write a Python script heavily utilising the re library for manipulation with

[1] The dependency of the tool on space characters separating words is a prerequisite for a broad coverage of languages/writing systems but rules out applicability to processing text in languages such as Chinese, Korean, Japanese and Thai. We have to employ other tools made for the particular language/script in these cases to tokenise texts well.

[2] The stream operating feature is necessary for making the tool a part of a corpus processing tool pipeline: the source plain text goes in, each processing tool is applied in a defined order to the result of the previous tool, the fully processed (tokenised, tagged, annotated, etc. data comes out.

[3] Reconstructing the original text can be useful for applying the tokenisation again after improving the tokenisation rules. Other tokenisers we use do not offer this option, therefore a copy of the original plain text has to be kept along with the tokenised vertical.

[4] http://unicode.org, the character class mapping is published at http://www.unicode.org/Public/UNIDATA/UnicodeData.txt (accessed 2014-11-17).

[5] A fast lexical analyzer, http://flex.sourceforge.net/

regular expressions[6]. The reason is we wanted to deal with some practical issues (described later on in this chapter) programmatically rather than by string/regexp matching and in the way which is more familiar to us.

We would like to acknowledge Laurent Pointal's Tree Tagger wrapper[7] which contributed the regular experessions to match the web related tokens. Unlike the Tree Tagger wrapper, `unitok` does just the tokenisation. It works independently of a particular tagger and thus can be combined with any tagger operating on text streams. The subsequent tagging, if required, is defined by the whole text processing pipeline.

3.2 The Method

As has been stated, `unitok` comes as a self standing Python script utilising the `re` library and operating on a text stream. The input text is decoded to unicode, normalised, scanned for sequences forming tokens, the tokens are separated by line breaks and the result vertical (one token per line) is encoded into the original encoding.

The normalisation deals with SGML entities by replacing them with unicode equivalents (e.g. '*&amp;*' by '*&*' or '*&ndash;*' by '*–*'). Unimportant control characters (the 'C' class in the Unicode specification) are stripped off. All whitespace (the 'S' class, e.g. a zero width space or a paragraph separator) is replaced by a single ordinary space.

Sequences of letters (i.e. words) are kept together. Sequences of numbers, marks, punctuation, and symbols kept by the normalisation are clustered together. The present SGML markup (usually HTML/XML) is preserved. The script also handles web related tokens: URLs, e-mail addreses, DNS domains, IP addresses are recognized. General abbreviations (uppercase characters optionally combined with numbers) are recognized.

Predefined language specific rules are applied. These are recognizing clitics (e.g. '*d'accord*' in French)[8], matching abbreviations (e.g. '*Prof.*' generally, '*např.*' in Czech)[9]) or special character rules (e.g. Unicode positions from 0780 to 07bf are considered word characters in Maldivian script Thaana).

The reversibility of tokenisation is ensured by inserting a 'glue' XML element between tokens not separated by a space in the input data. The most common case is the punctuation. An example showing the placement of the glue tag is given by Figure 1. The vertical with glue tags can be reverted to the original plain text using a short accompanying script `vert2plain`.

The language specific data (clitics, abbreviations, special word characters) are currently available for Czech, Danish, Dutch, English, French, Finnish,

[6] `https://docs.python.org/2/library/re.html`

[7] `http://perso.limsi.fr/pointal/dev:treetaggerwrapper`

[8] Lists of clictics were taken from the TreeTagger: `http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/`

[9] The lists of language specific abbreviations were taken from the respective Wikipedia page, e.g. `http://cs.wiktionary.org/wiki/Kategorie:%C4%8Cesk%C3%A9_zkratky` for Czech.

```
The
"
<g/>
end
<g/>
"
<g/>
.
```

Fig. 1: Example of tokenised and verticalised string *'The "end".'* showing the placement of the glue tag <g/>.

German, Greek, Hindi, Italian, Maldivian, Spanish, Swedish, Yoruba. Texts in other language are processed with default setting (making e.g. the *'prof.'* abbreviation always recognizable).

The output can be tagged (or generally further processed) by any tool operating on text streams with one token per line and XML tags.

4 Evaluation

A comparison of the output token count and the speed (expressed in output tokens per second) of three tokenisers – `unitok`, *TreeTaggerWrapper* and *Freeling* – can be found in Table 1. Two measurements were carried out using 1 GB sized plain texts in six European languages on a single core of Intel Xeon CPU E5-2650 v2 2.60 GHz. Specific recognition of clitics and abbreviations was on for all except Russian where the general setting was in effect. Only languages supported by TreeTaggerWrapper and Freeling were included in the test.

We found there is a noticeable difference between the tools in the number of output tokens. The speed tests revealed `unitok` was the slowest of the three tools but still quite sufficient for fast processing of large text data. Part of the performance drop is caused by providing the glue marks.

5 Conclusion

`unitok` is a fast processing tool for tokenisation of texts with spaces between words. The main benefits are good coverage of various sequences of characters, especially web phenomena, normalisation of messy control or whitespace characters, reversibility of the tokenised output and extensibility by language specific rules.

The tool has been successfully used for tokenising source texts for building large web corpora. The evaluation suggests we should consider improving the script to increase the speed of tokenisation in the future.

Acknowledgements This work has been partly supported by the Ministry of Education of CR within the LINDAT-Clarin project LM2010013 and by the Czech-Norwegian Research Programme within the HaBiT Project 7F14047.

Table 1: Comparison of tokenising speed and token count of *unitok*, *Freeling* and *TreeTaggerWrapper*. unitok is the base of the relative token count and the relative tokenising speed.

Language	tool	output tokens	rel tok	duration	tok/s	rel tok/s
English	Unitok	207,806,261	100%	6,880 s	30,200	100%
	TTWrapper	200,122,178	−3.70%	2,380 s	84,100	+178%
	Freeling	215,790,562	+3.84%	2,670 s	80,800	+168%
Spanish	Unitok	196,385,184	100%	6,250 s	31,400	100%
	TTWrapper	204,867,056	+4.32%	2,260 s	90,600	+188%
	Freeling	201,413,272	+2.56%	2,040 s	98,700	+214%
German	Unitok	171,354,427	100%	5,530 s	31,000	100%
	TTWrapper	179,120,243	+4.53%	2,360 s	75,900	+145%
French	Unitok	202,542,294	100%	6,400 s	31,600	100%
	TTWrapper	242,965,328	+20.0%	2,870 s	84,700	+168%
	Freeling	211,517,995	+4.43%	2,300 s	92,000	+191%
Russian	Unitok	98,343,308	100%	3,170 s	31,023	100%
	Freeling	102,565,908	+4.29%	1,450 s	70,800	+128%
Czech	Unitok	183,986,726		5,960 s	30,900	

References

1. Vít Baisa, Vít Suchomel, et al. Large corpora for turkic languages and unsupervised morphological analysis. In *Proceedings of the Eighth conference on International Language Resources and Evaluation (LREC'12), Istanbul, Turkey. European Language Resources Association (ELRA)*, 2012.
2. Gregory Grefenstette and Pasi Tapanainen. *What is a word, what is a sentence?: Problems of Tokenisation*. Rank Xerox Research Centre, 1994.
3. Miloš Jakubíček, Adam Kilgarriff, Vojtěch Kovář, Pavel Rychlỳ, Vít Suchomel, et al. The tenten corpus family. In *Proc. Int. Conf. on Corpus Linguistics*, 2013.
4. Adam Kilgarriff and Gregory Grefenstette. Introduction to the special issue on the web as corpus. *Computational linguistics*, 29(3):333–347, 2003.
5. David D Palmer. Tokenisation and sentence segmentation. *Handbook of natural language processing*, pages 11–35, 2000.
6. Vít Suchomel. Recent czech web corpora. In Pavel Rychlý Aleš Horák, editor, *6th Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 77–83, Brno, 2012. Tribun EU.

Finding the Best Name for a Set of Words Automatically

Pavel Rychlý

Natural Language Processing Centre
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
pary@fi.muni.cz

Abstract. Many natural language processing applicatons creates use clustering or other statistical methods to create sets of words. Such sets group together words with similar meaning and in many cases humans can find quickly an appropriate name. On the other hand computers represents such sets with a meaningless number or ID. This paper proposes an algorithm for automatic finding names of word sets. It provides examples of results as a simple evaluation of the methods.

Keywords: names of word sets, naming clusters, distributional thesaurus

1 Introduction

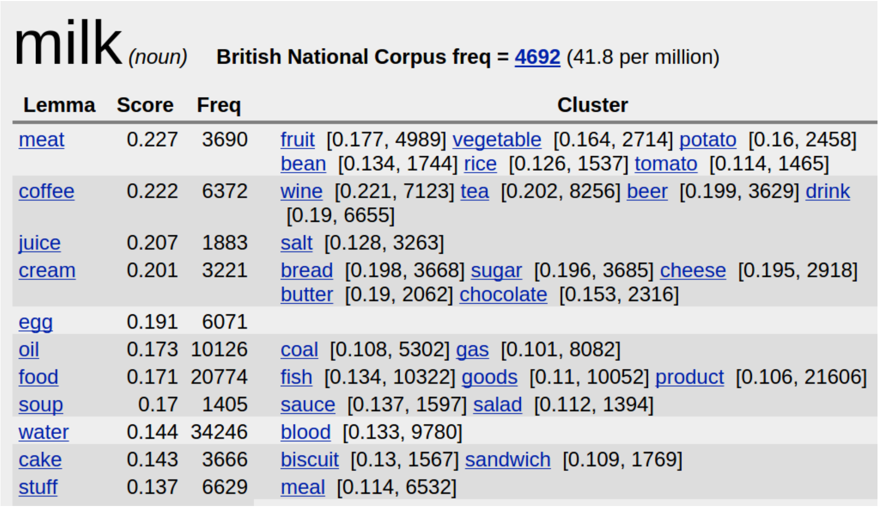
There are many applications in natural language processing which process words or lemmas and create some sets of words. Usually it is done via some type of clustering but they could be done using many different statistical methods.

As an example of such applications see Figure 1, it presents an thesaurus for word *milk* in the Sketch Engine system [1]. Thesaurus is computed automatically using a distributional similarity method [2]. The individual words which are similar to the given word (*milk*) are clustered using a bottom up clustering. The front words of each cluster is the word with the highest similarity score in the cluster.

The Sketch Engine thesaurus is based on the Word Sketches. These are one page collocational behavior of a word, an exampel of a Word Skech for verb *break* is displayed in Figure 2. In is used mainly in lexicography and language learning. A Word Sketch provides lists of collocations devided into several grammatical relations. On the Figure 2, some collocations are clustered using the same technique as in the Thesarus.

The final example is from LDA-frames project [3], Figure 3. LDA-frames is an unsupervised approach to identifying semantic frames from semantically unlabelled text corpora. There are many frame formalisms but most of them suffer from the problem that all frames must be created manually and the set of semantic roles must be predefined. The LDA-Frames approach, based on the

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2014, pp. 77–81, 2014. © Tribun EU 2014



milk *(noun)* **British National Corpus freq = 4692** (41.8 per million)

Lemma	Score	Freq	Cluster
meat	0.227	3690	fruit [0.177, 4989] vegetable [0.164, 2714] potato [0.16, 2458] bean [0.134, 1744] rice [0.126, 1537] tomato [0.114, 1465]
coffee	0.222	6372	wine [0.221, 7123] tea [0.202, 8256] beer [0.199, 3629] drink [0.19, 6655]
juice	0.207	1883	salt [0.128, 3263]
cream	0.201	3221	bread [0.198, 3668] sugar [0.196, 3685] cheese [0.195, 2918] butter [0.19, 2062] chocolate [0.153, 2316]
egg	0.191	6071	
oil	0.173	10126	coal [0.108, 5302] gas [0.101, 8082]
food	0.171	20774	fish [0.134, 10322] goods [0.11, 10052] product [0.106, 21606]
soup	0.17	1405	sauce [0.137, 1597] salad [0.112, 1394]
water	0.144	34246	blood [0.133, 9780]
cake	0.143	3666	biscuit [0.13, 1567] sandwich [0.109, 1769]
stuff	0.137	6629	meal [0.114, 6532]

Fig. 1: Thesaurus of *milk* in the Sketch Engine

Latent Dirichlet Allocation, avoids both these problems by employing statistics on a syntactically tagged corpus. The only information that should be given is a number of semantic frames and a number of semantic roles to be identified.

From all these examples we can see that many clusters clearly define one common meaning. A native speaker could easily choose a single word name for such cluster. This paper presents an algorithm to find such name automatically.

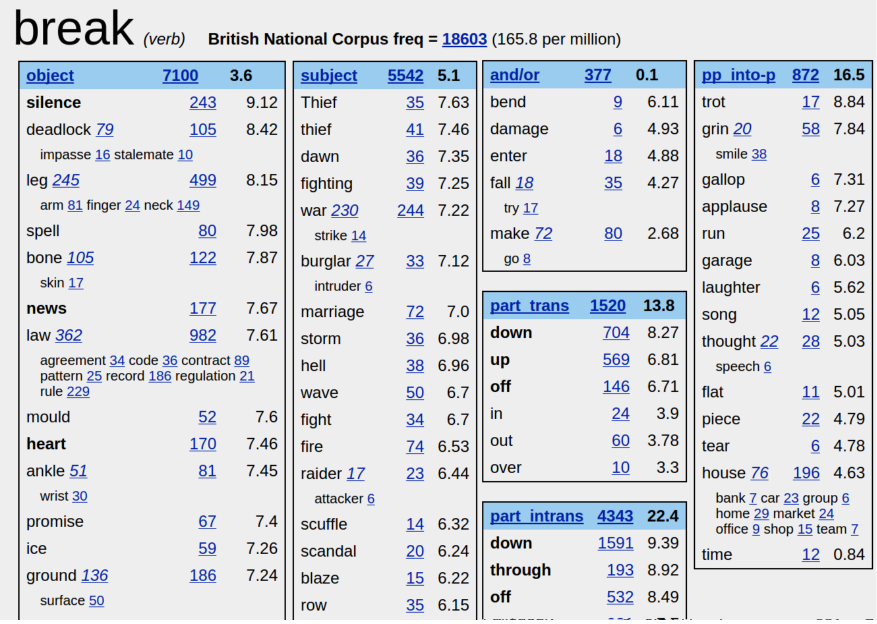
2 Proposed Method

The proposed method exploit the distributional thesaurus data which provide a list of similar words for a given word. The algorithm works as follows:

1. for each word in the given set find a list of top similar words in the thesaurus
2. sum the score for each of similar words across all given words
3. add 1 to the sums for each input words (the most similar word for any word is the word itself)
4. sort similar words according to the sums of scores
5. display the top items from the list

3 Evaluation

To our knowledge, there are no evaluation data available. We are going to prepare such gold data as a future work. As a simple form of evaluation we list results of the algorithm on our test data. They are presented in Table 1.



break (verb) **British National Corpus freq =** 18603 (165.8 per million)

object	7100	3.6
silence	243	9.12
deadlock 79	105	8.42
impasse 16 stalemate 10		
leg 245	499	8.15
arm 81 finger 24 neck 149		
spell	80	7.98
bone 105	122	7.87
skin 17		
news	177	7.67
law 362	982	7.61
agreement 34 code 36 contract 89 pattern 25 record 186 regulation 21 rule 229		
mould	52	7.6
heart	170	7.46
ankle 51	81	7.45
wrist 30		
promise	67	7.4
ice	59	7.26
ground 136	186	7.24
surface 50		

subject	5542	5.1
Thief	35	7.63
thief	41	7.46
dawn	36	7.35
fighting	39	7.25
war 230	244	7.22
strike 14		
burglar 27	33	7.12
intruder 6		
marriage	72	7.0
storm	36	6.98
hell	38	6.96
wave	50	6.7
fight	34	6.7
fire	74	6.53
raider 17	23	6.44
attacker 6		
scuffle	14	6.32
scandal	20	6.24
blaze	15	6.22
row	35	6.15

and/or	377	0.1
bend	9	6.11
damage	6	4.93
enter	18	4.88
fall 18	35	4.27
try 17		
make 72	80	2.68
go 8		

part_trans	1520	13.8
down	704	8.27
up	569	6.81
off	146	6.71
in	24	3.9
out	60	3.78
over	10	3.3

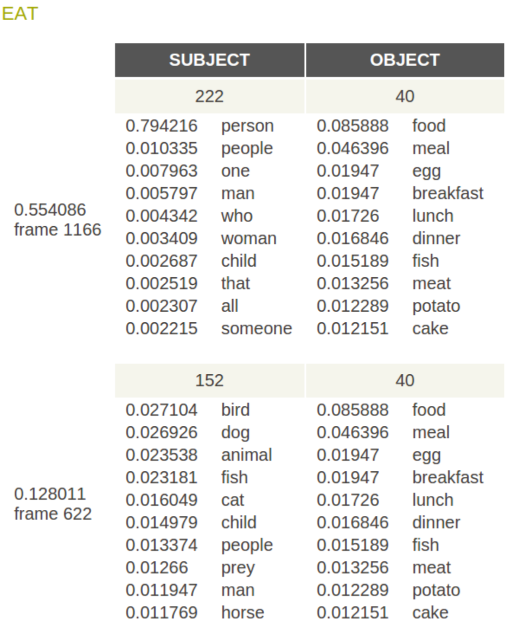
part_intrans	4343	22.4
down	1591	9.39
through	193	8.92
off	532	8.49

pp_into-p	872	16.5
trot	17	8.84
grin 20	58	7.84
smile 38		
gallop	6	7.31
applause	8	7.27
run	25	6.2
garage	8	6.03
laughter	6	5.62
song	12	5.05
thought 22	28	5.03
speech 6		
flat	11	5.01
piece	22	4.79
tear	6	4.78
house 76	196	4.63
bank 7 car 23 group 6 home 29 market 24 office 9 shop 15 team 7		
time	12	0.84

Fig. 2: Word sketch of verb *break* in the Sketch Engine

Table 1: Result of the algorithm on test data.

input word set	output top names
oil coal gas	fuel-n 0.696 energy-n 0.536
Britain Scotland Europe England	country-n 4.189 area-n 3.308
apple pear orange	fruit-n 2.145 thing-n 1.441
procedure study analysis method programme	system-n 5.367 work-n 4.959
pint bottle litre gallon	glass-n 2.371 water-n 2.258
meat fruit vegetable potato	food-n 3.291 fish-n 2.803
village town	city-n 0.611 area-n 0.478



EAT

	SUBJECT		OBJECT	
	222		40	
0.554086 frame 1166	0.794216	person	0.085888	food
	0.010335	people	0.046396	meal
	0.007963	one	0.01947	egg
	0.005797	man	0.01947	breakfast
	0.004342	who	0.01726	lunch
	0.003409	woman	0.016846	dinner
	0.002687	child	0.015189	fish
	0.002519	that	0.013256	meat
	0.002307	all	0.012289	potato
	0.002215	someone	0.012151	cake
	152		40	
0.128011 frame 622	0.027104	bird	0.085888	food
	0.026926	dog	0.046396	meal
	0.023538	animal	0.01947	egg
	0.023181	fish	0.01947	breakfast
	0.016049	cat	0.01726	lunch
	0.014979	child	0.016846	dinner
	0.013374	people	0.015189	fish
	0.01266	prey	0.013256	meat
	0.011947	man	0.012289	potato
	0.011769	horse	0.012151	cake

Fig. 3: Verb *eat* in LDA-frames

4 Interface

The algorithm is implemented as a command line cript. It is written in Python and use the Sketch Engine API to access the thesaurus data. We hope that after more finetuning the algorithm will be included into the Sketch Engine system. An example of a usage is at Figure 4.

```
$ clustname.py bnc2 bnc-hyper n Britain Scotland Europe England
country-n 4.18891489506
area-n 3.50870908797
year-n 3.5038651228
London-n 3.2635447681
world-n 3.13785666227
```

Fig. 4: An example of the clustname.py tool usage.

5 Conclusions

We have proposed an algorithm for finding names for a set of words. The implementation is mostly language and corpus independent and works quite well for many test data.

Acknowledgements This work has been partly supported by the Ministry of Education of CR within the LINDAT-Clarin project LM2010013 and by the Czech-Norwegian Research Programme within the HaBiT Project 7F14047.

References

1. Kilgarriff, A., Rychlý, P., Smrž, P., Tugwell, D.: The Sketch Engine. Proceedings of Euralex (2004) 105–116 `http://www.sketchengine.co.uk`.
2. Rychlỳ, P., Kilgarriff, A.: An efficient algorithm for building a distributional thesaurus (and other sketch engine developments). In: Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions, Association for Computational Linguistics (2007) 41–44
3. Materna, J.: LDA-Frames: An Unsupervised Approach to Generating Semantic Frames. In Gelbukh, A., ed.: Proceedings of the 13th International Conference CICLing 2012, Part I. Volume 7181 of Lecture Notes in Computer Science., Springer Berlin / Heidelberg (2012) 376–387

Part III

Semantics and Information Retrieval

Style Markers Based on Stop-word List

Jan Rygl and Marek Medved'

Natural Language Processing Centre
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
{rygl, xmedved1}@fi.muni.cz

Abstract. The analysis of author's characteristic writing style and vocabulary has been used to uncover the identity of authors of documents by both manual linguistic approaches and automatic algorithmic methods. The revealing of the gender, name, or age can help to expose pedophiles in social networks, false product reviews on the Internet servers, or machine translations submitted as manually translated texts.
These problems are predominantly solved by a combination of stylometry and machine learning techniques. Since the stylometry focuses on the author's style, word *n*-grams cannot be used as a style marker. Stop words are not influenced by a topic of documents, therefore they can be used to create style markers.
In this paper, we present a guidance on how to implement stop-word extraction and to include stop-words based style markers into a multilingual classification system based on the stylometry.

Keywords: style marker, stop-word list, corpus

1 Introduction

Anonymity is seen as the cornerstone of an Internet culture that promotes sharing and free speech. However, anonymity can also lead to crime. The uncovering of the true gender, name, or age of document authors can help to expose pedophiles in social networks, false product reviews on the Internet servers, or machine translations submitted as manually translated texts.

To reveal true identity of the document author, a variety of style markers have been used, with better or worse results. Style markers are documents features describing style and vocabulary of the author [1].

There are many stylometric features, such as word and sentence length, typography errors, vocabulary richness, punctuation marks, n-grams of syntactic labels, . . . [2,3,4,5]

In the first place, a vector of at least 100 most frequent words was used in a vast majority of recent approaches [6]. In the absence of stop-word sources, this paper provides a technical report how to generate list of stop words and implement style markers based on stop-word list including recommendation of tools for text preprocessing.

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2014, pp. 85–89, 2014. © Tribun EU 2014

2 Text preprocessing

To extract stop words from texts, we need to preprocess documents. Therefore, we advise to create a program for document preprocessing that takes raw text or HTML document as an input and outputs document objects that consist of:

1. raw source document
2. document language
3. character set that is used in the document
4. plain text without any HTML tags except a paragraph tag and a link tag where a diacritic check is provided on plain text's words
5. tokenization of the plain text
6. morphological annotation of the tokenized text
7. lemmatization of the tokenized text

The language of input HTML text can be determined by **langid** tool (for more information see [7,8]) that takes a text as a input and returns a language code in ISO 639-1 standard. Than the character set is derived from the input HTML text and it's language by **Chared** [9] (developed at Masaryk University in Brno). Information about the language of a document increases an accuracy of encoding detecton, therefore we recommend to do language detection before character set recognition.

Next we use `lxml.html.clean` from Python library (other tools depending on your programming language can be used) and get rid of all HTML tags except paragraph tags and links which can be useful for other style markers. In this step we also process all plain text's words by **czaccent** [10] (also developed at Masaryk University in Brno) tool that provide completion of diacritics if a word is spelled without or with incorrect diacritics. This process is necessary only for languages using characters with diacritics.

In the following step, the text is tokenized. We recommend to use **Unitok** [11] (universal tokenizing) program developed at Masaryk University in Brno that splits text into tokens and add predefined XML-like tags:

- `<doc>` – beginning of the document
- `<s>` – beginning of the sentence
- `</g>` – omitted space between tokens
- ...

After tokenization we pass the output into **Desamb** [12] tool (morphological desambiguator). The desamb uses morphological analyzer **Majka** [13] to annotate each word by morphological categories from the tagset of Majka and by lemmas. Then the best fitting variant of morphological category and lemma is selected for each token.

For our purposes we also train the **Majka** analyzer on Czech, Slovak and English data, thus our system can operate with this three languages. The output of desambiguation consists of morphological tags and lemmas for each token in the text. Lemmas statistics can be used instead of tokens to create a stop-word list used for style markers.

The whole process is illustrated in Figure 1.

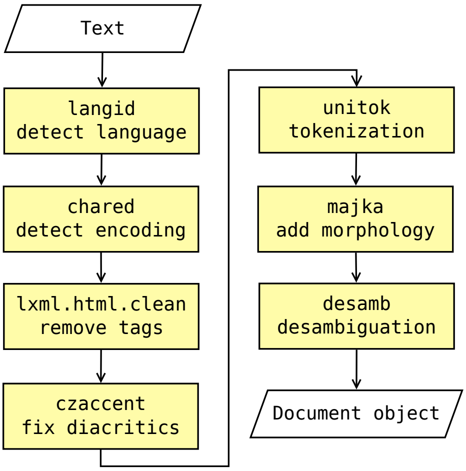


Fig. 1: Preprocessing text

3 Stop-word style markers

The purpose of this characteristic is to find the most frequent words (stop words) of given language in the text and provide style marker values for them.

3.1 Stop-word extraction from a corpus

To obtain list of stop words of a given language, **Sketch engine**'s [14,15] `freq` tool can be used on large corpora. We extracted stop words for Czech, Slovak and English documents:

- for Czech stop words we use `czTenTen` corpus that consists of 5 069 447 935 tokens
- for Slovak stop words we use `skTenTen` corpus that consists of 876 003 720 tokens
- for English stop words we use `enTenTen` corpus that consists of 12 968 375 937 tokens

To extract stop-word lists using Sketch engine, you can use two console commands:

```
command:
        freqs $corpora '[]' 'word 0 tag 0' > word_feq
        freqs $corpora '[]' 'lemma 0 tag 0' > lemma_feq
```

where `$corpora` is a name of the corpus. We used following corpora:

- sk: sktenten

- cs: cztenten12_8
- en: ententen12

The advantage of the Sketch engine is that is already contains corpora for almost every world language. If you do not have access to the Sketch engine and you have own corpora, you can generate stop words with frequencies using bash command line tools or any programming language.

3.2 Style markers extraction

This characteristic can use two input types thus user can choose if the calculations operate with lemmas or tokens (*word* represents a lemma or a token). Below we describe calculations on words but the same calculations can be provided on lemmas too.

We calculate two types of frequencies: the absolute frequency and the relative frequency of stop words that appear in the input text. This characteristic is used to generate three different predominant outputs:

1. relative frequencies of stop words:

$$\frac{count_{stop\_word}(document)}{count_{word}(document)}$$

2. absolute differencies of absolute values of relative frequencies of words from the corpus and relative frequencies of words in the input text

$$\left|\frac{count_{stop\_word}(corpus)}{count_{word}(corpus)}\right| - \left|\frac{count_{stop\_word}(document)}{count_{word}(document)}\right|$$

3. squared values of differencies of absolute values of relative frequencies of words from the corpus and relative frequencies of words in the input text

$$\left(\frac{count_{stop\_word}(corpus)}{count_{word}(corpus)} - \frac{count_{stop\_word}(document)}{count_{word}(document)}\right)^2$$

The first method ignores corpus frequencies and it is suitable for scenarios we are given stop-word lists, but frequencies are counted from an untrustworthy corpus or are unknown. For other situations, we recommend other two variants. The third variant is sensitive to big deviations from corpus frequencies which can be important style marker.

4 Conclusions

The style markers based on stop-word lists are easily implemented. To generate own style markers we recommend to use mentioned methods or their alternatives for other programming languages. Style markers based on stop-word lists are still considered to be very beneficial for the most of algorithms using stylometry, therefore each software solving these problems should implement them.

Acknowledgements This work has been partly supported by the Ministry of Education of CR within the LINDAT-Clarin project LM2010013.

References

1. Stamatatos, E., Fakotakis, N., Kokkinakis, G.: Automatic text categorization in terms of genre and author. Computational Linguistics **26**(4) (2000) 471–495
2. Mosteller, F., Wallace, D.L.: Inference and Disputed Authorship: The Federalist. Addison-Wesley (1964)
3. Holmes, D.I.: Authorship attribution. Literary and Linguistic Computing **13**(3) (1998) 111–117
4. Juola, P.: Authorship Attribution. Foundations and Trends in Information Retrieval **1** (2006) 233–334
5. Grieve, J.: Quantitative authorship attribution: An evaluation of techniques. Literary and Linguistic Computing **22**(3) (2007) 251–270
6. Eder, M.: Style-markers in authorship attribution a cross-language study of the authorial fingerprint. Studies in Polish Linguistics **2011**(Volume 6 Issue 1) (2011)
7. Lui, M., Baldwin, T.: Cross-domain feature selection for language identification. In: In Proceedings of 5th International Joint Conference on Natural Language Processing. (2011) 553–561
8. Lui, M., Baldwin, T.: Langid.py: An off-the-shelf language identification tool. In: Proceedings of the ACL 2012 System Demonstrations. ACL '12, Stroudsburg, PA, USA, Association for Computational Linguistics (2012) 25–30
9. Pomikálek, Jan and Suchomel, Vít: Chared: Character Encoding Detection with a Known Language. In: Aleš Horák, Pavel Rychlý. RASLAN 2011, Tribun EU, 5th ed. Brno (Czech Republic) (2011) 125–129
10. Rychlý, P.: CzAccent - Simple Tool for Restoring Accents in Czech Texts. In Horák, A., Rychlý, P., eds.: 6th Workshop on Recent Advances in Slavonic Natural Language Processing, Brno, Tribun EU (2012) 15–22
11. Michelfeit, J., Pomikálek, J., Suchomel, V.: Text Tokenisation Using unitok. In Horák, A., Rychlý, P., eds.: 8th Workshop on Recent Advances in Slavonic Natural Language Processing, Brno, Tribun EU (2014) 71–75
12. Šmerk, P.: K počítačové morfologické analýze češtiny (in Czech, Towards Computational Morphological Analysis of Czech). PhD thesis, Faculty of Informatics, Masaryk University (2010)
13. Šmerk, P.: Fast morphological analysis of czech. In: Proceedings of Third Workshop on Recent Advances in Slavonic Natural Language Processing, Brno, Tribun EU (2009) 13–16
14. Kilgarriff, A., Baisa, V., Bušta, J., Jakubíček, M., Kovář, V., Michelfeit, J., Rychlý, P., Suchomel, V.: The sketch engine: ten years on. Lexicography **1**(1) (2014) 7–36
15. Kilgarriff, A., Baisa, V., Bušta, J., Jakubíček, M., Kovář, V., Michelfeit, J., Rychlý, P., Suchomel, V.: Statistics used in the sketch engine. (2014)

Separating Named Entities

Barbora Ulipová and Marek Grác

Computational Linguistics Centre
Faculty of Arts
Masaryk University
Czech Republic
b.ulipova@gmail.com, grac@mail.muni.cz

Abstract. In this paper, we analyze the situation of long sequences of mostly capitalized words which look like a named entity but in fact they consist of several named entities. An example of such phenomena is *hokejista (hockey player) New York Rangers Jaromír Jágr*. Without splitting the sequence correctly, we will wrongly assume that the whole capitalized sequence is a name of the hockey player. To find out how the sequence should be split into the correct named entities, we tested several methods. These methods are based on the frequencies of the words they consist of and their n-grams. The method DIFF-2 proposed in this article obtained much better results than MI-score or logDice.

Keywords: text corpus, mutual information, named entities

In the process of creating a new question answering system, we have encountered several problems. One of the problems that has to be solved is extracting the roles, titles or occupation of various people from a free text. This information together with (at least part of the) name of person often occur together and it should be possible to extract them. In this paper we will focus on a situation when a longer capitalized sequence of words is found and it is required to split the sequence into multiple parts. In the next section we will describe examples found during research in Czech corpora. The third section describes possible approaches based on frequencies and co-occurrences of words. The results are presented in the last section of this paper.

1 Named entities characterization

In this project, we focus on extracting two types of information. First one, identify person's features such as role, title or occupation of person. In that case we are looking for a list of words that can be prepared in advance. Such words are *actress (herečka)*, *model (modelka)*, *hockey player (hokejista)*, *minister (minister)*. These words are often mentioned in tabloids that are a great resource of such information. The second information type we are looking for is a relationship between two people where both of them are mentioned. Often, they are not referenced by full names but only by the first name or a nickname, e.g. *Brooklyn,*

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2014, pp. 91–96, 2014. © Tribun EU 2014

son of David Beckham.This information is sometimes set, sometimes it can change in time (e.g. *wife*, *boss*). The Process of extracting valid information for given time is not a part of this paper.

In the Czech, another issue occurs regularly. In the corpus, we can find clusters of capitalized words where parts of such clusters belong to separate named entities. An example of such phrase is *hráč New York Rangers Jaromír Jágr (New York Rangers' player Jaromír Jágr)* or *hra Vladimíra Franze Válka s mloky (Vladimír Franz's play Válka s mloky)*. These phrases have to be identified and separated into the relevant parts.

In this paper we focus on methods to split such sentences of mostly capitalized words into multiple parts. During our research we have not met many sequences that require to be split to more than two parts, so we will focus mainly on the ones which only need to be split into two parts in next sections. The other cases can be solved by running our methods multiple times.

2 Extraction of names

The first step in creating a system for extraction of information about people is finding the names of people. People's names are, obviously, capitalized in a text which we can use. However, there are other words which are capitalized as well – proper names such as names of institutions, nationalities, products and artistic works. Therefore, we need to find a way to distinguish between people and other capitalized words. We have a list of nouns that can represent a person [1] and also a list of words that describe a relationship between people. Our preliminary research shows that if we look for the relationship words in the vicinity of two or more capitalized words which are not at the beginning of a sentence, the majority of capitalized words found will contain names of persons.

3 Separating the phrases

To separate the phrases, we have decided to use an approach based on frequencies and co-occurrences of words. There are several ways how to express a co-occurrence between words, we have tried the MI-score [2] and the logDice [3]. In this section, we will present various methods together with examples on real sequences obtained from corpus czTenTen12_8 [4]. At first, we tried to write corpus queries in CQL [5] but we have found out that one complex query finished in minutes which makes it almost unusable in the real-world applications. We had to simplify the queries to work just with the n-grams which occur in a corpora that can be found in a different way based on the preprocessed data and now a query took less than 0.1 seconds on the same hardware.

3.1 Method based on mutual information between two words

The first, most naive approach is based on counting MI-score or logDice between bigrams in sequence. The sequence is a segment of text that should be divided into separate named entitities. Let's have a $sequence = (w_1, w_2, w_3, ..., w_X)$ then we will count both logDice and MI-score for each pair (w_I, w_{I+1}). According to the theory of mutual information, it can be expected that lower values of mutual score provide a better identification of the borders as these words should be the words that relate to each other less than others. If the sequence itself or candidate n-gram divisions are not mentioned in corpus, we have to modify the equation a bit and replace the occurrences of zero by one. Our test suite[1] showed that using logDice results had the precision **29.5%** while MI score's precision was only **11.8%**. Examples of selected sequences are shown in table 1 where the text in bold represents the correct division and the numbers in bold represent the results according to used methods.

Table 1: Detailed results of using method based on MI between two words. Sequences shown in table are *PSP Miroslava Němcová* and *D. Cerekve Zdeněk Jirsa*

	MI-score	logDice		MI-score	logDice
PSP - Miroslava	**8.77**	**-14.26**	D. - Cerekve	**-0.49**	-18.18
Miroslava - Němcová	10.21	-13.71	**Cerekve - Zdeněk**	1.08	-18.19
			Zdeněk - Jirsa	2.69	**-19.44**

3.2 Method based on mutual information between n-grams

In order to improve this method, we have decided to extend the above mentioned equations to work directly on n-grams. There will be only two n-grams for each division because we want to split the sequence to just two parts as was explained before. The first n-gram will start at the first word of the sequence and the second one will end with the last word. Such n-grams do not have to be a part of the corpora so we will reuse the modification proposed in the previous subsection and we will replace zeros with ones. Such modifications will also be a part of every other proposed method.

When we use this method, we can expect that higher values of mutual score will represent the correct division. This should happen because the mutual information between two incorrectly selected n-grams should be lower as they do not occur together as frequently as the correct ones. The precision results when logDice was used is **41.2%** and MI score's precision is **29.5%**.

[1] Test suite was manually created from 17 phrases that were selected according to estimation of frequency of different types in corpus

Table 2: Detailed results of using method based on MI between n-grams.

	MI-score	logDice
PSP - Miroslava Němcová	10.07	-11.23
PSP Miroslava - Němcová	**10.89**	**-9.86**
D. - Cerekve Zdeněk Jirsa	-1.49	-13.83
D. Cerekve - Zdeněk Jirsa	**10.64**	**-4.30**
D. Cerekve Zdeněk - Jirsa	4.44	-8.01

3.3 Method extended with negative n-grams

The previous method did not take into account that if a particular word always follows the n-gram, it should be a part of it, too. Our next method is based on very same idea. In order to count only the *clean* value we will subtract the number of longer n-grams from the frequency of the shorter one. Because we are working with the n-gram model, finding negative occurrences is not possible like it is in CQL itself, so we have simplified this process to testing the first adjacent word to our n-gram. For example when if we have a sequence *New York Rangers Jaromír Jágr* then we will count the modified frequency for n-gram *New York* as follows: *freq(New York) - freq(New York Rangers)*.

In the ideal case, the modified frequency will be zero which will in the case of this sequence happen only if there is not any other *Jaromír* in this hockey team as he will have different surname (last word of sequence). In such case the modified frequency will be the frequency of occurrences of the same sequence with just the last word replaced. The number subtracted from frequency provides additional information that can be used for evaluation too.

Table 3: Detailed results of using method based on MI between n-grams with modified frequencies

	MI-score	logDice
PSP - Miroslava Němcová	10.17	**-10.62**
PSP Miroslava - Němcová	**11.19**	N/A
D. - Cerekve Zdeněk Jirsa	-1.49	-13.62
D. Cerekve - Zdeněk Jirsa	**11.19**	**-3.26**
D. Cerekve Zdeněk - Jirsa	4.46	N/A

When modified frequencies will be used in the n-gram equations, the results show that the are too many results in intermediate steps that can not be counted as the modified frequency drops to zero. As the results were heavily corrupted by this, we have decided not to count the mutual information of n-grams but to use only the value subtracted in equations. We have created two equations that make sense to us.

The first one, is based on the idea that when we sum modified frequencies (*DIFF-4* – use four elements in equation) of n-grams the lowest value will show the ideal borderline between them. The other one is just simplification of previous one where we will sum together just subtracted values (*DIFF-2* – use two elements in equation). Unlike the previous methods these numbers are not normalized at all but they work correctly on a selected sequence. In order to compare values across the sequences, some normalization will have to be done which will be a part of following research.

According to our test suite, the *DIFF-4* has a precision **35.2%** which ranks it above all usage of the MI-score which could not be expected before the experiment. The *DIFF-2* method has a precision **94.1%** with only one error on the test suite. This example is *D. Cerekve Zdeněk Jirsa* where borders were properly identified by n-gram methods based on both MI-score and logDice. When we expand *D.* to *Dolní* then division will be found correctly.

4 Results and Future Work

In this paper, we have presented several methods that should divide a sequence of words into two semantically correct parts. According to our results, using bigram model does not provide the best precision on the test suite. Extending equations for MI-score and logDice rapidly increases obtained precision. It was shown that in both of these cases using logDice resulted into improved precision against MI-score, so we can suggest to test it also for other applications.

The best method is DIFF-2 with the precision **94.1%**. It is surprising that it does not take into account the frequency of the final n-grams. One of the reasons why they are not used yet is that the results are not normalized so they cannot be compared across various sequences like MI-score and logDice allows. Obtained results are not really representative yet, so we will prepare a larger dataset for Czech.

Although people tend to split such phrases quite easily, it is often a result of detecting patterns of either n-grams or word-class like first names. We did not use first names or any other source of information to be able to improve our methods in the real applications.

In the future, we would like to focus on a way to detect sequences that have to divided. Also the idea of normalization of DIFF-2 and DIFF-4 is very interesting as then they can also be tested against MI-score and logDice in different applications and for different languages.

Acknowledgements This work has been partly supported by the Masaryk University within the project *Čeština v jednotě synchronie a diachronie – 2014* (MUNI/A/0792/2013).

References

1. Grác, M.: Rapid Development of Language Resources. PhD thesis, Dissertation, Masaryk University in Brno (2013)
2. Church, K.W., Hanks, P.: Word association norms, mutual information, and lexicography. Computational linguistics **16**(1) (1990) 22–29
3. Rychlý, P.: A lexicographer-friendly association score. Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN (2008) 6–9
4. Jakubíček, M., Kilgarriff, A., Kovář, V., Rychlý, P., Suchomel, V., et al.: The tenten corpus family. In: Proc. Int. Conf. on Corpus Linguistics. (2013)
5. Arasu, A., Babu, S., Widom, J.: Cql: A language for continuous queries over streams and relations. In: Database Programming Languages, Springer (2004) 1–19

Intelligent Search and Replace for Czech Phrases

Zuzana Nevěřilová and Vít Suchomel

Natural Language Processing Centre
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
{xpopelk,xsuchom2}@fi.muni.cz

Abstract. This work proposes a new improvement of the 'Search and Replace' function well known from most text processing software.
The standard search and replace function is used to replace exact form of words or phrases by another words or phrases in text documents. It is quite sufficient for languages with minimal inflection such as English. However, a well working word or phrase replacement function for morphologically rich languages requires much more thought.
We explore the issues of implementing a useful search and replace in the Czech language and propose solutions to majority of the problems: A syntactic parser is employed to identify the phrases containing the search word or phrase. The correct word forms used as a replacement are generated by a morphological analyser.
A web demonstration utilizing the proposed solution is presented. The attached examples of use reveal the cases in which the implemented method works well.

Keywords: search and replace, detecting phrases, generating phrases, subject-predicative complement

1 Motivation

This work introduces an intelligent search and replace editing tool for a text processor in Czech. The basic search and replace is a well known function implemented by most text processing software. A search word (or a phrase) and the new text to replace by is entered by the user. The text processor finds all occurrences of the search string and performs the replacement. Exact matches are made therefore the user has to know exact forms of the search phrase. (Some tools support searching regular expressions but that does not offer any grammatical advantage over the basic method.)

The basic function is quite sufficient for languages with minimal inflection such as English. Nevertheless, global search and replace without additional knowledge can cause problems (see e.g. the Scunthorpe problem[1]).

We call our tool "intelligent" since it uses language knowledge to 1) search all occurences of a word regardless its forms, and 2) avoid searching

[1] See https://en.wikipedia.org/wiki/Scunthorpe_problem

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2014, pp. 97–105, 2014. © Tribun EU 2014

coincidentally equal substrings. Our work deals with the issue for Czech – a highly inflective language. The improved function should be able to find all inflected forms of the search phrase and apply the respective morphological forms to the replacement phrase. The new search and replace tool offers the ability to automate tedious manual edits like the basic function. A big concern should be placed in reaching a reasonable precision. The users would not tolerate it making mistakes.

Having such tool, one could automate many frequent cases of replacement to save editing time:

- correction of often repeated mistakes,
- unification of terms in translation (done by the chief translating editor),
- especially unification of terms in localisation (e.g. GUI descriptions: 'dialog box' – 'dialogové okno' (neuter gender), 'dialogový box' (masculine inanimate),
- adjusting general parts of manuals to particular products,
- changing ingredients in recipes,
- replacing person or company names in standardised documents,
- especially in legal text, e.g. common parts of contracts
- and other standardized labels, notices, signs.

2 Related Work

There is no work dealing with search and replace function of Czech phrases known to the authors in the Czech speaking environment. Although there is e.g. a Czech grammar checker available for Microsoft Word [6], arguably the most widespread text processor used for Czech, there is no module for search and replace available in the tool.

A general idea of morphological search and replace has been patented by Microsoft [8]. In addition to that, the approach presented in this paper covers also search and replace of multiword phrases and is able to deal with Czech phenomena that are uncommon to English – grammatical gender and subject-predicative complement agreements.

3 Methods

In order to replace whole phrases, we need to identify them in many different forms. For this reason, we use the syntactic parser SET [2]. For lemmatization and tagging, we use the tools `majka` and `desamb` respectively. The parser takes tagged vertical as input, separates individual phrases and determines their heads. It also determines the *phrase lemma* and *phrase tag*: in case of noun phrases, the phrase tag corresponds to the head tag; in case of prepositional phrases, it corresponds to the complement noun phrase. The phrase tag determination is crucial for the replacement method. The phrase lemma corresponds to the respective lemmata in the phrase but in addition, adjective, pronoun, and numeral modifiers are changed to fulfil the grammatical agreement. For example, the noun phrase “tato dvě červená jablka” (these two red apples) has the respective lemmata: “tento” (this), “dva” (two), “červený” (red), “jablko” (apple). The noun phrase is at the same time the noun phrase lemma since it is nominative.

If the replacement concerns head of a noun phrase or head of a complement noun phrase (in prepositional phrases), the head tag gender and number is compared to that of the new phrase. The replacement has to follow the same case and number as the original phrase tag. The gender can be different and if it is, the obligatory agreements have to be fulfilled.

In Czech, three basic types of grammatical agreement are related to noun phrases:

- head modifiers: this grammatical agreement does not occur solely in Slavonic languages, it is also known in Romance languages: adjectives, pronouns and numerals modifying a particular head have to agree in number and gender with the head
- active verb: the grammatical agreement between the subject and the verb phrase can be complicated for analytical verbs (e.g. past tense in Czech, that is composed from active verb *to be* in present tense and past participle), moreover it relies on a correct detection of the syntactic subject
- predicative complement: if the complement is an adjective, pronoun or numeral, it has an obligatory agreement in gender and number with the subject. In this case, the predicative complement is hard to detect (it depends on the copula verb occurence and the copula verbs are defined in a very arguable way).

More formally, the replacement of phrase p by r in text T ($p \to r$) proceeds in the following way:

1. detect phrase lemma $p(lemma)$ and phrase tag $p(tag)$ for search phrase p, and phrase lemma $r(lemma)$ and phrase tag $r(tag)$ for the replacement r
2. parse whole T, separate individual noun phrases and prepositional phrases, detect their phrase tags and phrase lemmata
3. if $p(lemma)$ is found in i-th phrase in T:
   (a) replace $p_i(lemma)$ with $r(lemma)$, we label this particular replacement by the same index: r_i
   (b) modify gender of all adjective, pronoun, and numeral modifiers of $r_i(lemma)$ if the genders of $p(tag)$ and $r(tag)$ differ.
   (c) if $p_i(tag)$ is not nominative, decline $r_i(lemma)$ according to the case of $p_i(tag)$
   (d) if r_i is part of the subject and the clause contains a copula verb, modify the predicative complement according to the gender of $r_i(tag)$ (only adjective, pronoun, and numeral predicative complements are subject of gender agreement)
   (e) if r_i is part of the subject and the verb phrase contains a participle, modify the verb participle according to the gender of $r_i(tag)$

For all mentioned types of inflection, we use the declension and conjugation tool [5] based on morphological generator majka [7]. Examples of grammatical agreements follow:

Adjective Modifiers vysoký dům → vysoká budova

Subject-Predicate dům stál na nabřeží → budova stála na nábřeží

Subject-Predicative Complement dům byl vysoký → budova byla vysoká

One imporant pitfall concerning the declension exists: the tagger can detect incorrectly the case or the gender of the noun phrase/prepositional phrase. For example, for the noun phrase "zahraniční podnik" (foreign enterprise) the case can be either nominative or accusative (the word forms are the same). In this case, the parsing passes without problems with both nominative or accusative but the quality of the tagging has serious consequences. If we replace "podnik" (enterprise, in Czech masculine inanimate) by "firma" (company, in Czech feminine), the resulting forms differ depending on the case ("firma" in nominative and "firmu" in accusative). In addition, other sentence parts (verb phrase or predicative complement) may or may not change (depending on whether the phrase is a syntactic subject).

The replacement is therefore "intelligent" in the sense that it replaces noun phrases and prepositional phrases not substrings. Nevertheless, is does not take word senses into consideration. This can be an issue when the assumption *one sense per discourse* is not fulfilled. Particularly, in case of phrases that are part of phrasemes, the replacement can lead to incorrect results. For example, imagine replacing "hand" by "foot". In this case, the phrase "on the other hand" will result into "on the other foot". Replacing all occurences regardless the context can lead into errors, however, [1] proved that in 98% cases, the assumption *one sense per discourse* is correct.

4 Results

A web demonstration utilizing the proposed solution for Czech has been made available at `http://nlp.fi.muni.cz/projects/phrase_replace`. We have used the application to perform replacements in instruction manuals, cooking recipes and a definition page from an encyclopedia. The method works well in these cases since there are not many complicated sentences there and the present tense is usually used. See the appendix for comparison of input and output example texts representing these kinds of replacements.

We also tried general replacements of single words as well as multi word phrases. Despite the tool made mistakes outlined in the previous chapter, it performed fairly well in two thirds of cases: 17 of 30 instances of changing "dům" to "stavba" in 25 random sentences from a Czech web corpus were completely right, 4 replaces introduced a small error but the declension of the target word was right (e.g. ″v stavbě“ instead of ″ve stavbě“) and 9 mistakes were made in declension or in recognising the right part of speech (e.g. ″domů“ can be a noun or an adverb). Pronominal anaphora resolution proved to be an issue in longer and complex sentences.

The accuracy of the method is not perfect. The problematic coverage of the anaphora related issues or word sense desambiguation makes it even harder. Therefore we recommend to ask the user to confirm each replacement of the search phrase and allow a manual edit at key places in the case of utilising the tool in a word processor.

5 Conclusion and Future Work

The paper presents a preliminary work concerning inflection-aware search and replace of phrases in Czech. It seems that not many previous projects pursue this topic not only in Czech but also in other languages. The reason is that a successful replacement tool depends on other NLP tasks. In the current version, we employ morphological analysis, tagging, parsing, and morphological generation for inflection of the replaced phrases as well as phrases that are subject of obligatory agreements. However, the tool is not aware of co-references in the text. Future work will therefore concern three main subtasks of co-reference resolution:

- **Zero subject resolution** In Slavonic languages, the subject does not have to be expressed in each sentence. Zero subject resolution is needed in sentences containing past participles in verb phrases or predicative complements. It is solved e.g. by [4] but not yet used in our tool. Zero subject resolution is useful for replacements such as Bob → Alice as subject. For example: Bob se narodil v Brně, kde také vystudoval. (Bob is born in Brno where he also studied.) Alice se narodila v Brně, kde také *vystudovala*. (Alice is born in Brno where she also *studied*.)
- **Pronominal anaphora resolution** Resolution of possessive, personal, and demonstrative pronouns seems to be a simpler task in morphologically rich languages with several grammatical agreements than in English. It is partially solved by [4,3], however not yet implemented in our tool. Pronominal anaphora resolution is suitable for replacements such as batoh → taška (backpack → bag). For example: Rozdělal jsem přezky na batohu a začal vybalovat věci. Najednou z *něj* vypadl plátěný pytlík s něčím omamně voňavým. (I opened the backpack and started to unpack it. Suddenly, a small canvas sack with something perfumed dropped out of *it*.)
- **Abbreviated forms** Even in technical text where synonymy is not desired, abbreviated forms exist. To our knowledge, no language tool for recognizing abbreviated forms of noun phrases exist. In future, we need to build such tool, probably based on similarity search and/or corpus-based thesauri. Abbreviated forms could correctly replace phrases such as: Elektrické travní sekačky jsou ideální na udržování menších travnatých ploch.

> Vyžadujete-li tichý chod a ohleduplnost k životnímu prostředí, vyberte si *elektrickou sekačku*! (Electric lawn mowers are suitable for maintenance of smaller areas. If you require silent operation and environment considerations, choose an *electric mower*!)

Acknowledgements This work has been partly supported by the Ministry of Education of CR within the LINDAT-Clarin project LM2010013 and by the Czech-Norwegian Research Programme within the HaBiT Project 7F14047.

References

1. William A. Gale, Kenneth W. Church, and David Yarowsky. One sense per discourse. In *Proceedings of the Workshop on Speech and Natural Language*, HLT '91, pages 233–237, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics.
2. V. Kovář, A. Horák, and M. Jakubíček. Syntactic analysis using finite patterns: A new parsing system for Czech. In *Human Language Technology. Challenges for Computer Science and Linguistics*, volume November 6–8, 2009, pages 161–171, Poznań, Poland, 2011.
3. Lucie Kučová and Zdeněk Žabokrtský. Anaphora in czech: Large data and experiments with automatic anaphora resolution. In Václav Matoušek, Pavel Mautner, and Tomáš Pavelka, editors, *Proceedings of 8th International Conference on Text, Speech and Dialogue, TSD 2005*, volume 3658 of *Lecture Notes in Computer Science*, pages 93–98. Springer Berlin Heidelberg, 2005.
4. Václav Němčík. Saara: Anaphora resolution on free text in Czech. In Aleš Horák and Pavel Rychlý, editors, *Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2012*, pages 3–8, Brno, Czech Republic, 2012. Tribun EU.
5. Zuzana Nevěřilová. Declension of Czech noun phrases. In Jan Radimský, editor, *Actes du 31e Colloque International sur le Lexique et la Grammaire*, pages 134–138, České Budějovice, 2012. Université de Bohême du Sud à České Budějovice (République tchèque).
6. K. Oliva, V. Petkevič, and Microsoft s.r.o. Czech grammar checker, 2005.
7. Pavel Šmerk. Unsupervised learning of rules for morphological disambiguation. In Petr Sojka, Ivan Kopeček, and Karel Pala, editors, *TSD*, volume 3206 of *Lecture Notes in Computer Science*, pages 211–216. Springer, 2004.
8. J. E. Walsh and R. A. Fein. Morphological search and replace, 02 1999.

Appendix: Example inputs (left column) and outputs (right column) of the intelligent search and replace tool

Example 1: Instruction manual of a tool. Replace *kráječ → bazuka*.

Kráječ na potraviny je zařízení, které doma můžete používat na krájení potravin na tenké plátky jako z lahůdkářství. Tyto domácí kráječe fungují stejným způsobem jako komerční, avšak nejsou tak výkonné. Kráječ na potraviny Vám umožní nakrájet maso a sýry na požadovanou tloušťku. Protože každý kráječ je trochu jiný, podívejte se do manuálu na specifickou konstrukci Vašeho kráječe. Většinu kráječů koupíte z velké části složenou. Postavte základnu kráječe na neklouzavý povrch. Vyberte si místo, kde budete mít dost prostoru na práci s kráječem, avšak kde nikomu nebude překážet a kde se nikdo nezraní o jeho velmi ostrý kotouč.

Bazuka na potraviny je zařízení, které doma můžete používat na krájení potravin na tenké plátky jako z lahůdkářství. Tyto domácí *bazuky* fungují stejným způsobem jako komerční, avšak nejsou tak *výkonné*. *Bazuka* na potraviny Vám umožní nakrájet maso a sýry na požadovanou tloušťku. Protože *každá bazuka* je trochu *jiná*, podívejte se do manuálu na specifickou konstrukci *vaší bazuky*. Většinu *bazuk* koupíte z velké části *složenou*. Postavte základnu *bazuky* na neklouzavý povrch. Vyberte si místo, kde budete mít dost prostoru na práci s *bazukou*, avšak kde nikomu nebude překážet a kde se nikdo nezraní o *jeho* velmi ostrý kotouč.

Example 2: A cooking recipe. Replace *prsíčka → kýta*.

Troubu předehřejte na 190 °C. Kůži na prsíčkách dobře propíchejte vidličkou, přelijte vroucí vodou a pak nechte dobře oschnout. V těžké, silnostěnné nepřilnavé pánvi pomalu opékejte prsíčka nejdříve kůží dolů. Pak otočte a nechte opéci z druhé strany. Vyjměte z pánve a osolte. Jablka rozložte do lehce olejem vytřené zapékací formy nebo pekáčku. Posypte tymiánem, přidejte svitek skořice a navrch rozložte opečená prsíčka kůží nahoru.

Troubu předehřejte na 190 °C. Kůži na *kýtách* dobře propíchejte vidličkou, přelijte vroucí vodou a pak nechte dobře oschnout. V těžké, silnostěnné nepřilnavé pánvi pomalu opékejte *kýty* nejdříve kůží dolů. Pak otočte a nechte opéci z druhé strany. Vyjměte z pánve a osolte. Jablka rozložte do lehce olejem vytřené zapékací formy nebo pekáčku. Posypte tymiánem, přidejte svitek skořice a navrch rozložte *opečené kýty* kůží nahoru.

Example 3: A definition page from an encyclopedia. Replace *ptakoještěr* → *slepice*.

Pterodactylus („křídelní prst“) byl rod pterodaktyloidního ptakoještěra žijícího na území dnešního Německa a zřejmě i jinde v Evropě a Africe v období svrchní jury (asi před 151–148 miliony let). Tento létající plaz je jedním z prvních objevených a vědecky popsaných ptakoještěrů vůbec. První zkameněliny byly identifikovány již roku 1784 Cosimou A. Collinim. Řádně vědecky popsán pak byl počátkem 19. století. Byl to dravec, který se živil zejména rybami a jinými malými obratlovci, případně i bezobratlými. K lovu mu sloužily také drobné zuby na okrajích čelistí. Rozpětí křídel pterodaktylů dosahovalo jen kolem 1,5 metru u dospělých exemplářů, patřil tedy mezi menší ptakoještěry.[2]

Pterodactylus („křídelní prst“) byl rod *pterodaktyloidního slepice* žijícího na území dnešního Německa a zřejmě i jinde v Evropě a Africe v období svrchní jury (asi před 151–148 miliony let). Tento létající plaz je jedním z prvních objevených a vědecky popsaných *slepic* vůbec. První zkameněliny byly identifikovány již roku 1784 Cosimou A. Collinim. Řádně vědecky *popsán* pak *byl* počátkem 19. století. *Byl* to dravec, který se živil zejména rybami a jinými malými obratlovci, případně i bezobratlými. K lovu mu sloužily také drobné zuby na okrajích čelistí. Rozpětí křídel pterodaktylů dosahovalo jen kolem 1,5 metru u dospělých exemplářů, patřil tedy mezi menší *slepice*.

Example 4: Random sentences from the Czech web. Replace *dům* → *stavba*.

Věděla, že se v hořícím domě nachází člověk neschopný se vlastními silami dostat ven. Každý dům má padacími dveřmi chráněný vchod obrácený na jih a malá okénka tam, kde zeď domu převyšuje střechu domu sousedního. Každý zákazník má již od začátku stavby jasný manuál co vše má dům obsahovat a může si vše lehce kontrolovat.

Věděla, že se v *hořící stavbě* nachází člověk neschopný se vlastními silami dostat ven. *Každá stavba* má padacími dveřmi chráněný vchod obrácený na jih a malá okénka tam, kde zeď *stavby* převyšuje střechu *stavby sousedního*. Každý zákazník má již od začátku stavby jasný manuál co vše má *stavbu* obsahovat a může si vše lehce kontrolovat.

[2] From Czech Wikipedia: `http://cs.wikipedia.org/wiki/Pterodactylus`.

Example 5: A name phrase. Replace *chráněná krajinná oblast → národní park*.

Budeme se snažit vyhlásit chráněnou krajinnou oblast Prameny Ploučnice. Je nejrozsáhlejší chráněnou krajinnou oblastí v České republice, nabízející množství přírodních rezervací. Jako nejrozsáhlejší chráněná krajinná oblast v České republice nabízela množství přírodních rezervací.

Budeme se snažit vyhlásit *národní park* Prameny Ploučnice. Je *nejrozsáhlejším národním parkem* v České republice, *nabízející* množství přírodních rezervací. Jako *nejrozsáhlejší národní park* v České republice *nabízel* množství přírodních rezervací.

Example 6: A change from a male name to a female name. Replace *Václav Havel → Eva Nováková*.

Václav Havel působil v 60. letech 20. století v Divadle Na zábradlí, kde jej také proslavily hry Zahradní slavnost (1963) a Vyrozumění (1965). 9. července 1964 se Václav Havel po osmileté známosti oženil s Olgou Šplíchalovou. Po vypuknutí Sametové revoluce v listopadu 1989 se Václav Havel stal jedním ze spoluzakladatelů protikomunistického hnutí Občanské fórum a jako jeho kandidát byl 29. prosince 1989 zvolen prezidentem Československa.[3]

Eva Nováková působila v 60. letech 20. století v Divadle Na zábradlí, kde *jej* také proslavily hry Zahradní slavnost (1963) a Vyrozumění (1965). 9. července 1964 se *Eva Nováková* po osmileté známosti *oženila* s Olgou Šplíchalovou. Po vypuknutí Sametové revoluce v listopadu 1989 se *Eva Nováková stala jedna* ze spoluzakladatelů protikomunistického hnutí Občanské fórum a jako jeho *kandidát byl* 29. prosince 1989 *zvolen prezidentem* Československa.

[3] From Czech Wikipedia: `http://cs.wikipedia.org/wiki/V%C3%A1clav_Havel`.

An Architecture for Scientific Document Retrieval
Using Textual and Math Entailment Modules

Partha Pakray and Petr Sojka

Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
{pakray,sojka}@fi.muni.cz

Abstract. We present an architecture for scientific document retrieval. An existing system for textual and math-ware retrieval Math Indexer and Searcher MIaS is designed for extensions by modules for textual and math-aware entailment. The goal is to increase quality of retrieval (precision and recall) by handling natural languge variations of expressing semantically the same in texts and/or formulae.
Entailment modules are designed to use several, ordered layers of processing on lexical, syntactic and semantic levels using natural language processing tools adapted for handling tree structures like mathematical formulae. If these tools are not able to decide on the entailment, generic knowledge databases are used deploying distributional semantics methods and tools. It is shown that sole use of distributional semantics for semantic textual entailment decisions on sentence level is surprisingly good. Finally, further research plans to deploy results in the digital mathematical libraries are outlined.

Keywords: math-aware information retrieval, semantic textual entailment, math entailment, distributional semantics, Gensim

1 Introduction

Semantic-based document filtering and search module is a key component of any Information Retrieval (IR) system. Search is a gateway to the ever-growing database of documents in digital libraries (DL) or on the web. Even though keyword based IR systems became part of everyday life today, they are not fully suitable for research search to DLs, for example. The more precise results the information seeker might get are those expressed, queried, indexed, and retrieved based on word, sentence, paragraph, or document *meaning*, e.g. semantic features of the document content.

The variation in expressivity of natural languages, including the mathematical vernacular, to describe semantically similar ideas and elements is enormous. Keyword-based information systems try to cope with it on lexical level by morphology (indexing lemmas) or by synonymical expansion like Wordnet. There is 'semantic web' and ontology-based approaches based on discrete, dichotomic representations of words and relations between them. But they are often not

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2014, pp. 107–117, 2014. © Tribun EU 2014

enough to handle and uniformly represent document, paragraph, sentence or formulae *meaning* in IR systems, e.g. for semantically fine-grained document filtering and similarity computations.

On the other hand, distributional semantic approaches have deserved well-grounded attention recently. They allow to represent word or phrase meaning in continuous high-dimensional spaces, just based on unsupervised, and often deep, learning methods [15]. Such representations can be used for purposes like qualified guesses of semantic similarity of words, phrases, or even sentences or formulae.

In this paper, we design an extension module for our math-aware information system MIaS [21]. We argue that it will further increase current performance [12,20] by better, semantic clustering of variably expressed content.

The motivation for new architecture design is discussed in Section 2. We describe how distributional semantics may help to compute semanticaly similar text chunks or formulae. In Section 3 the new entailment modules of the architecture are described. We conclude by Section 4 by describing further directions of research.

2 Motivation for a New Architecture

When checking precision of MIaS on results from [12,20], we have realized that some documents are not found just because of minor rephrasing of formulae or text in query with respect to the document. We need a *robust* way of computing *similarity* for textual phrases and formulae terms. In STEM papers, the text is full of formulae, where we cannot simply discard them as they convey very important semantics in dense form: *semantic* textual similarity is needed.

2.1 Semantic Textual Similarity

The main goal of Semantic Textual Similarity (STS) task [1] is measuring the degree of semantic equivalence between a pair of texts, e.g. sentences. This task can applied in many areas as Information Extraction, Question Answering, Summarization and in Information Retrieval area for indexing the semantically same phrases or sentences. Three STS evaluation tasks were organised in 2012 [3], 2013 [2], and 2014 [1] at SemEval workshops. In that evaluation tasks, the systems performance was evaluated using the Pearson product-moment correlation coefficient between the participant system scores and the human scores.

Textual similarity problem may be tackled by various techniques at lexical, syntactic and semantic levels 1, as usual during NLP processing. Among lexical techniques there are word overlap metrics or n-gram matching. Another way is to compare dependency relations of two texts. In computations one can use synonyms, hypernyms, etc. The higher processing level, the better performance is usually achieved.

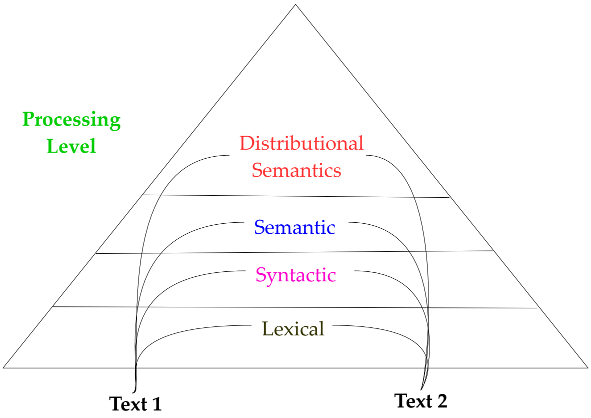


Fig. 1: Natural language processing levels

There always remain some examples which cannot be decided by lexical, syntactic nor semantical analysis, as full knowledge and meaning representation is needed for it. There is *semantic gap* between lexical surface of the text and its meaning because same concepts are represented in different vocabulary, languages, formalisms and notations. Updating knowledge databases with all dialectical possibilities in supervised way is doomed to failure.

In *distributional semantics* approaches [5], similarities between linguistic items could be computed from their collocativity and distributional properties in large samples of language data in unsupervised way, as clearly seen from visualization experiments [7]. Especially convincing are recent experiments computed by *`Gensim`* framework [18] where words and phrases are computed by `Word2vec` [14] language model. We have tried to use it for STS task.

2.2 Sentence Level Similarity Baseline Experiment

Our STS system will generate various kinds of features from each processing level as shown in Figure 1. Finally, it will use machine learning to decide on the similarity between two text chunks as shown in later on Figure 3 on page 113.

In a preliminary experiment we have used already pre-trained word and phrase vectors available as part of Google News dataset [14] (about 100 billion words). The LSA word-vector mappings model contains 300-dimensional vectors for 3 million words and phrases.

`Gensim` [18] is a Python framework for vector space modelling. We have used `Gensim` for this experiment, and computed the cosine distance between vectors representing text chunks – sentences from SemEval tasks.

We have used English test data of Sematic Textual Similarity (STS) Task 6 [3] from SemEval-2012, Task 6 [2] from SemEval-2013, Task 10 [1] from SemEval-2014. Given two snippets of text, STS measures their degree of semantic equivalence. The SemEval organizers provided English sentence pairs of news headlines (corpus named HDL), pairs of glosses (OnWN), image descriptions (Images), DEFT-related discussion forums (Deft-forum) and news (Deft-news), and tweet comments and newswire headline mappings (Tweets).

Table 1: SemEval-2014 Task 10: Multilingual Semantic Textual Similarity Test Result

Corpus	Winner score and team/run name		Our score
Deft-forum	0.5305	NTNU-run3	0.42812
Deft-news	0.7850	Meerakat_mafia-Hulk	0.67999
Headlines	0.7837	NTNU-run3	0.60985
Images	0.8343	NTNU-run3	0.71402
OnWN	0.8745	MeerkatMafia-paringWords	0.79135
Tweet-news	0.7610	DLS@CU-run1	0.76571

Table 2: SemEval-2013 Task 6: Semantic Textual Similarity Test Result

Corpus	Winner score and team/run name		Our score
Headlines	0.7838	UMBC_EBIQUITY-saiyan	0.62501
OnWN	0.8431	deft-baseline	0.71165
FNWN	0.5818	UMBC_EBIQUITY-ParingWords	0.38353
SMT	0.6181	UMBC_EBIQUITY-ParingWords	0.32951

Table 3: SemEval-2012 Task6: Semantic Textual Similarity Test Result

Corpus	Winner score and team/run name		Our score
MSRpar	0.6830	baer/task6-UKP-run2_plus_postprocessing_smt_twsi	0.30103
MSRvid	0.8803	jan_snajder/task6-takelab-simple	0.68318
SMT-europal	0.5581	sranjans/task6-sranjans-1	0.54057
On-WN	0.7273	weiweitask6-weiwei-run1	0.68779
SMT-news	0.6085	desouzatask6-FBK-run3	0.51915

Tables 1, 2, and 3 show results of our minimalistic system based on distributional semantics language model compared to highest sentence similarity scores of systems participating in SemEval-2014, 2013 and 2012. It is worth noting that for Tweet-news subtask at SemEval-2014 our 'baseline' system using only plain `Word2vec` with pretrained Google news data by LSA gave *better result than the best system* at SemEval-2014!

Just recently, another way of computing *global* distributional semantics has been reported by Stanford's `GloVe` [16]. We will compare its performance with `Word2vec`. As our results on SemEval data indicate that training corpora is very important, we have realized that Wikipedia knowledge to tackle the

STS Similarity problem is crucial, including the named entities and formulae available there.

2.3 Learning from Wikipedia Corpus

Wikipedia is an online encyclopedia that contains millions of articles on a wide variety of topics with quality comparable to that of traditional encyclopedias. In [22,17,23], Wikipedia has been used as a successful measure of semantic relatedness between words or text passages.

We will build word and phrase vectors from Wikipedia articles[1]. This Wikipedia dump contains more than 3 billion words. We will use `Word2vec` for learning high-quality word vectors from Wikipedia data sets with billions of words. An example for vector representation could be as follows: vector("King") − vector("Man") + vector("Woman") results in a vector that is closest to the vector representation of the word Queen. [15]

We will test on SemEval STS test data by using this generated vector from Wikipedia articles. Finally, we will compare our results with our baseline system. We will also participate in STS evaluation track at SemEval 2015 Task 2[2]. Having good similarity measures on scientific text chunks, we may use it for our math-aware information retrieval system.

3 New MIaS Architecture with Entailment Modules

Our top-level system architecture is shown in Figure 2. The architecture used sofar is enriched by three modules: Text-Text Entailment (TE), Math-Math Entailment (ME) and Text-Math Entailment (TME) modules.

Textual entailment is defined in [9] as: text T is said to entail hypothesis H if the truth of H can be inferred from T. The task of Textual entailment is to decide whether the meaning of H can be inferred from the meaning of the T.

For example, the text T = "John's assassin is in jail" entails the hypothesis H = "John is dead"; indeed, if there exists one's assassin, then this person is dead. On the other hand, T = "Mary lives in Europe" does not entail H = "Mary lives in US". Much effort is devoted by the Natural Language Processing (NLP) community to develop advanced methodologies in TE which is considered as a core NLP task. Various international conferences and several evaluation track competitions on TE have been held, notably at PASCAL-Pattern Analysis, Statistical Modelling and Computational Learning[3], Text Analysis Conferences (TAC)[4] organized by the United States National Institute of Standards and Technology (NIST), Evaluation Exercises on Semantic Evaluation (SemEval)[5], National Institute of Informatics Test Collection for Information Retrieval

[1] `http://dumps.wikimedia.org/enwiki/`

[2] `http://alt.qcri.org/semeval2015/task2/`

[3] `http://pascallin.ecs.soton.ac.uk/Challenges/`

[4] `http://www.nist.gov/tac/tracks/index.html`

[5] `http://semeval2.fbk.eu/semeval2.php`

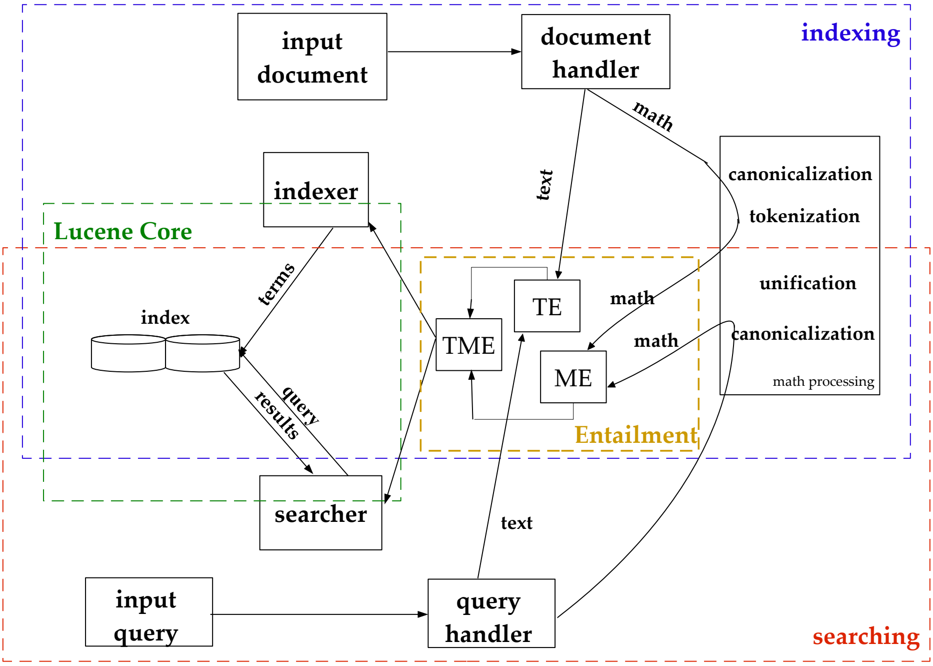


Fig. 2: Scheme of the new MIaS system workflow, enriched by entailment modules

System (NTCIR)[6] since 2005. At each new TE competition, the participating teams introduced several new features in their TE systems ranging from lexical to syntactic to semantic methodologies from two-way (i.e. binary-class) to multi-way (i.e. multi-class) textual entailment classifications in monolingual to cross-lingual scenario in order to solve the TE problem.

In this work we will investigate into the use of entailment modules for IR. We will show that Textual and Math entailment plays a significant role for monolingual IR performance.

The general architecture of Textual Entailment system is shown in Figure 3 on the next page. Text and Hypothesis comparison is represented by comparative analysis; and the entailment decision is made by a classifier that makes use of a feature vector.

The Textual Entailment system is unidirectional but Semantic Textual Similarity is mainly bidirectional. Table 4 on page 115 shows our system result of Semantic Textual Similarity and compare to the Entailment.

In the MIaS system [21] search can be done by three ways e.g. only text search, only mathematics formula search and text with mathematics formula

[6] http://research.nii.ac.jp/ntcir/

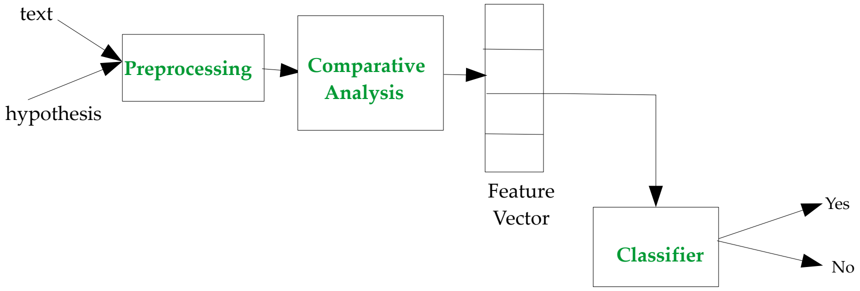


Fig. 3: General Textual Entailment architecture

search. During the searching phase, a query can match several terms in the index. However, one match can be more important to the query than another, and the system must consider this information when scoring matched documents. An example of TE module is shown in Figure 4 on the next page.

ME module will compare between Math query and document that contained math formula. For example, $x^2 + y^2 = z^2$ entails $a^2 + b^2 = c^2$. We will implement *Math Entailment* in Formulae weighting module [21]. We will try to use Math Entailment module in this phase to find appropriate terms. An example of the ME module is shown in Figure 5 on the following page.

TME Module will compare text and math within documents. TME module not only increases fairness of similarity ranking, but also helps to match a query against the indexed form by adding new terms for indexing, e.g. formulae for named entity used to name it. TME module is shown in Figures 4 and 5 on the next page.

Entailment module will search not only for whole sentences (whole formulae), but also for single words and phrases (subformulae down to single variables, symbols, constants, etc.). For calculating the relevance of the matched expressions to the user's query, entailment module will use a matching technique of indexed mathematical terms, which accordingly affects scores of matched documents and thus the order of results.

In our TE system based on lexical similarity we will determine the similarity between the two texts by our STS module. Additionally, we will compare the dependency structure between the two texts.

The TE problem can be tackled by various ways like lexical, syntactic and semantic. Sometimes lexical semantic similarity is not sufficient to solve the TE problem. In Table 1, for pair Id 5 our lexical semantic similarity system have given high score of 0.95 but the meaning of text1 and text2 is very different. In this case dependency structure weighting verb as main decision factor may solve the problem.

Tree structure of input sentences are widely used by many research groups, since it provides more information with quite good robustness and runtime than shallow parsing techniques. Basically, a dependency parsing tree contains

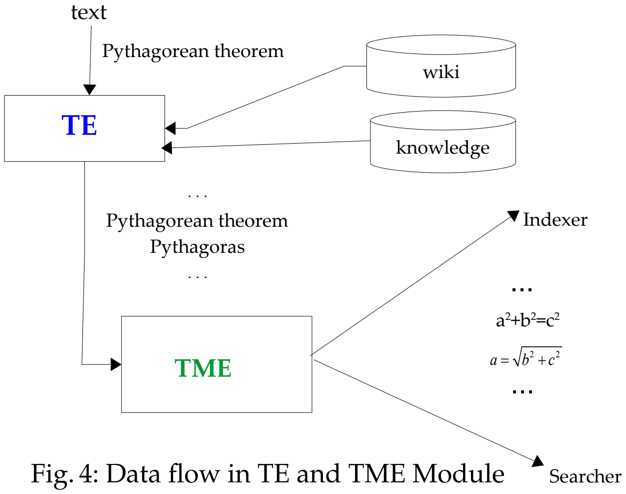


Fig. 4: Data flow in TE and TME Module

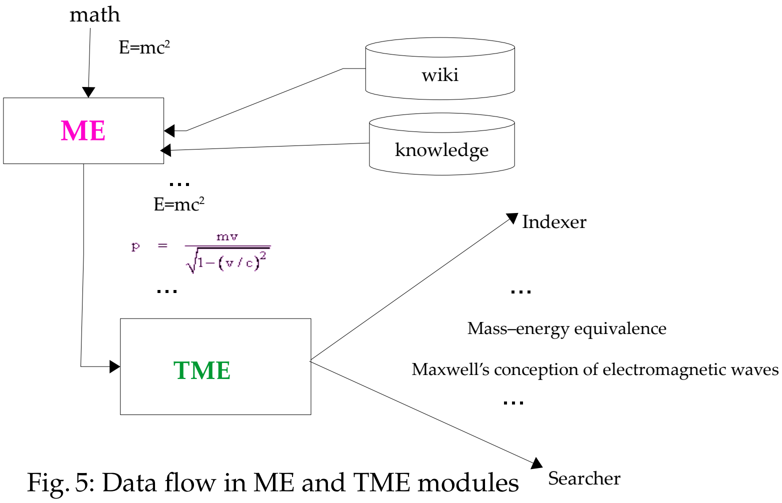


Fig. 5: Data flow in ME and TME modules

nodes (i.e., tokens/words) and dependency relations between nodes. Some approaches simply treat it as a graph and calculate the similarity between the text and the hypothesis graphs solely based on their nodes, while some others put more emphasis on the dependency relations themselves. The recent approaches of syntactic or tree edit models are [10,13,19]. The approach in [11] based on the tree edit distance algorithm, which contains three basic operators, insertion, deletion and substitution. Insertion is defined as the insertion of a node from the dependency tree of *H* into the dependency tree of *T*; deletion is the removal of a node from the dependency tree of *T*, together with all its attached children; and substitution is the change of the label of a node

Table 4: Example of pairs from Task 1 at SemEval 2014

Id	Text 1	Text 2	our STS	Entailment
1	One young boy is climbing a wall made of rock	A young child is climbing a rock climbing wall which is indoors	0.7871	No
2	A man is phoning	A man is talking on the phone	0.8238	Yes
3	John was born on January 15, 1986 in Kolkata.	John was born in 1986 in the city of Kolkata.	0.7996	No
4	A woman is performing a trick on a ramp with a bicycle	A woman is jumping with a bicycle	0.7839	No
5	A brown dog is attacking another animal in front of the man in pants	A brown dog is helping another animal in front of the man in pants	0.95	No

in the source tree (the dependency tree of *T*) into a label of a node of the target tree (the dependency tree of *H*). Substitution is allowed only if the two nodes share the same part-of-speech (POS). The approach in [4] presents a new data structure, termed compact forest, which allows efficient generation and representation of entailed consequents, each represented as a parse tree. Rule-based inference is complemented with a new approximate matching measure inspired by tree kernels, which is computed efficiently over compact forests. The approach [24] built a model to solve the entailment problem by using dependency syntax analysis (by Stanford Parser), lexical knowledge base (e.g. WordNet), web information (e.g. Wikipedia) and probabilistic methods.

We will generate dependency tree for two texts. Then mapping can be done in two ways e.g. directly (when entities from hypothesis dependency tree exist in the text tree) or indirectly (when entities from text tree or hypothesis tree cannot be mapped directly and need transformations using external resources). Based on this step we will decide on our entailment resulting implementation.

4 Conclusion and Further Work

We have described an architecture for math-aware information retrieval that employs textual and math entailment. We have described our further research directions: distributional approaches that we will test for entailment modules. We want also train distributional semantics representation for mathematical formulae, and test to which extent their vectors may be used to approximate their meaning. Finally, we plan to use SEPIA evaluation tool and NTCIR's Math task [12] data to evaluate the improvements, and eventually use it in the digital mathematics libraries as EuDML [6] or planned GDML [8].

Acknowledgement This work was supported by an ERCIM Alain Bensoussan Fellowship 2014–15 and Masaryk University. Any opinions, findings, and conclusions expressed here are those of the authors and do not necessarily reflect the view of the ERCIM or MU.

References

1. Agirre, E., Baneab, C., Cardiec, C., Cerd, D., Diabe, M., Gonzalez-Agirrea, A., Guof, W., Mihalcea, R., Rigau, G., Wiebeg, J.: Semeval-2014 task 10: Multilingual semantic textual similarity. In: Proceedings of SemEval 2014. p. 81 (2014)
2. Agirre, E., Cer, D., Diab, M., Gonzalez-Agirre, A., Guo, W.: Sem 2013 shared task: Semantic textual similarity including a pilot on typed-similarity.* sem 2013: The second joint conference on lexical and computational semantics. In: Association for Computational Linguistics (2013)
3. Agirre, E., Diab, M., Cer, D., Gonzalez-Agirre, A.: Semeval-2012 task 6: A pilot on semantic textual similarity. In: Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation. pp. 385–393. Association for Computational Linguistics (2012)
4. Bar-Haim, R., Berant, J., Dagan, I.: A compact forest for scalable inference over entailment and paraphrase rules. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing. vol. 3, pp. 1056–1065. Association for Computational Linguistics (2009)
5. Blei, D., Ng, A., Jordan, M.: Latent Dirichlet Allocation. The Journal of Machine Learning Research 3, 993–1022 (2003)
6. Borbinha, J., Bouche, T., Nowiński, A., Sojka, P.: Project EuDML—A First Year Demonstration. In: Davenport, J.H., Farmer, W.M., Urban, J., Rabe, F. (eds.) Intelligent Computer Mathematics. Proceedings of 18th Symposium, Calculemus 2011, and 10th International Conference, MKM 2011. Lecture Notes in Artificial Intelligence, LNAI, vol. 6824, pp. 281–284. Springer-Verlag, Berlin, Germany (Jul 2011), http://dx.doi.org/10.1007/978-3-642-22673-1_21
7. Chaney, A.J., Blei, D.M.: Visualizing topic models. In: International AAAI Conference on Social Media and Weblogs. Department of Computer Science, Princeton University, Princeton, NJ, USA (Mar 2012)
8. Cole, T.W., Daubechies, I., Carley, K.M., Klavans, J.L., LeCun, Y., Lesk, M., Lynch, C.A., Olver, P., Pitman, J., Xia, Z.J.: Developing a 21st Century Global Library for Mathematics Research. National Research Council, Washington, D.C.: The National Academies Press (Mar 2014)
9. Dagan, I., Glickman, O.: Probabilistic textual entailment: Generic applied modeling of language variability. In: Proceedings of PASCAL Workshop on Learning Methods for Text Understanding and Mining. p. 6. Grenoble (2004), http://u.cs.biu.ac.il/~dagan/publications/ProbabilisticTE_fv07.pdf
10. Heilman, M., Smith, N.A.: Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics. pp. 1011–1019. Association for Computational Linguistics (2010)
11. Kouylekov, M., Magnini, B.: Recognizing textual entailment with tree edit distance algorithms. In: Proceedings of the First Challenge Workshop Recognising Textual Entailment. pp. 17–20 (2005)
12. Líška, M., Sojka, P., Růžička, M.: Similarity Search for Mathematics: Masaryk University team at the NTCIR-10 Math Task. In: Kando, N., Kishida, K. (eds.) Proceedings of the 10th NTCIR Conference on Evaluation of Information Access Technologies. pp. 686–691. National Institute of Informatics, Tokyo, Japan (2013), http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings10/pdf/NTCIR/MATH/06-NTCIR10-MATH-LiskaM.pdf

13. Mai, Z., Zhang, Y., Ji, D.: Recognizing text entailment via syntactic tree matching. In: Proceedings of the 9th NII Test Collection for Information Retrieval Workshop (NTCIR '11). pp. 361–364 (2011)
14. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed Representations of Words and Phrases and their Compositionality. In: Burges, C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K. (eds.) Advances in Neural Information Processing Systems 26, pp. 3111–3119. Curran Associates, Inc. (2013), http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf
15. Mikolov, T., Yih, W., Zweig, G.: Linguistic Regularities in Continuous Space Word Representations. In: Proceedings of HLT-NAACL 2013. pp. 746–751 (2013)
16. Pennington, J., Socher, R., Manning, C.: Glove: Global Vectors for Word Representation. In: Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014). 12 pages (Oct 2014), http://nlp.stanford.edu/projects/glove/glove.pdf
17. Ramprasath, M., Hariharan, S.: Using ontology for measuring semantic similarity for question answering system. In: International Conference on Advanced Communication Control and Computing Technologies (ICACCCT). pp. 218–223. IEEE (Aug 2012), http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6320774
18. Řehůřek, R., Sojka, P.: Software Framework for Topic Modelling with Large Corpora. In: Proceedings of LREC 2010 workshop New Challenges for NLP Frameworks. pp. 45–50. ELRA, Valletta, Malta (May 2010), http://is.muni.cz/publication/884893/en, software available at http://nlp.fi.muni.cz/projekty/gensim
19. Rios, M., Gelbukh, A.: Recognizing textual entailment with a semantic edit distance metric. In: 11th Mexican International Conference on Artificial Intelligence (MICAI). pp. 15–20. IEEE (2012)
20. Růžička, M., Sojka, P., Líška, M.: Math Indexer and Searcher under the Hood: History and Development of a Winning Strategy. In: Joho, H., Kishida, K. (eds.) Proceedings of the 11th NTCIR Conference on Evaluation of Information Access Technologies. 8 pages. National Institute of Informatics, Tokyo, Japan (Dec 2014), https://is.muni.cz/auth/publication/1201956/en
21. Sojka, P., Líška, M.: The Art of Mathematics Retrieval. In: Proceedings of the ACM Conference on Document Engineering, DocEng 2011. pp. 57–60. Association of Computing Machinery, Mountain View, CA (Sep 2011), http://doi.acm.org/10.1145/2034691.2034703
22. Strube, M., Ponzetto, S.P.: WikiRelate! Computing semantic relatedness using Wikipedia. In: AAAI. vol. 6, pp. 1419–1424 (2006)
23. Witten, I., Milne, D.: An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In: Proceeding of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy, AAAI Press, Chicago, USA. pp. 25–30 (2008)
24. Xu, X., Wang, H.: ICL Participation at RTE-7. In: Proceedings of Text Analysis Conference TAC 2011. 4 pages. NIST, Gaithersberg, Maryland, USA (Nov 2011), http://www.nist.gov/tac/publications/2011/participant.papers/ICL.proceedings.pdf

Part IV

Morphology and Lexicon

SQAD: Simple Question Answering Database

Marek Medveď and Aleš Horák

Natural Language Processing Centre
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
{xmedved1,hales}@fi.muni.cz

Abstract. In this paper, we present a new free resource for comparable Czech question answering evaluation. The Simple Question Answering Database, SQAD, contains 3301 questions and answers extracted and processed from the Czech Wikipedia. The SQAD database was prepared with the aim of a precision evaluation of automatic question answering systems. Such resource was currently not available for the Czech language. We describe the process of SQAD creation, processing of the texts by automatic tokenization (Unitok) and morphological disambiguation (Desamb) and successive semi-automatic cleaning and post-processing.
We also show the results of a first version of Czech question answering system named SBQA (syntax-based question answering).

Keywords: question answering, Simple Question Answering Database, SQAD, syntax-based question answering, SBQA

1 Introduction

The question answering (QA) field has a significant potential nowadays. Systems that are capable of answering any possible question can be widely used by general public to answer questions about the weather, public transport etc. In specialized applications, question answering can be used in medicine to speed up the process of fitting the patient symptoms to all possible illnesses.

Nowadays, several perspective question answering systems appeared, such as the IBM's Watson [1,2] that had a big success in popular television competitive show Jeopardy![1]. Many QA systems are orientated on a specific domain and are always able to answer only simple questions [3,4,5,6].

Question answering systems employ tools that process the input question than go through a knowledge base and provide a reasonable answer to the question. The presented SQAD database will help to measure and improve accuracy of QA tools as it offers all relevant processing parts, i.e. the source text, the question and the expected answer.

[1] Jeopardy! is an American television show that features a quiz competition in which contestants are presented with general knowledge clues in the form of answers, and must phrase their responses in question form.

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2014, pp. 121–128, 2014. © Tribun EU 2014

Original text:

Létající jaguár je novela spisovatele Josefa Formánka z roku 2004.

[Létající jaguár is a novel of writer Josef Formánek form the 2004.]

Question:

Kdo je autorem novely Létající jaguár?

[Who is the author of the novel of Flying jaguar?]

Answer:

Josef Formánek

URL:

`http://cs.wikipedia.org/wiki/L%C3%A9taj%C3%ADc%C3%AD_jagu%C3%A1r`

Author:

chalupnikova

Fig. 1: Example of a `SQAD` record

2 The `SQAD` Database

The Simple Question Answering Database, `SQAD`, was created in a computational linguistic course by students. Their task was to choose sections from Czech Wikipedia articles as a source for different questions and the respective answers. `SQAD` contains 3301 records with the following data fields (see Figure 1 for an example):

- the original sentence(s) from Wikipedia
- a question that is directly answered in the text
- the expected answer to the question as it appears in the original text
- the URL of the Wikipedia web page from which the original text was extracted
- name of the author of this `SQAD` record

In the first phase, the `SQAD` database consisted of plain texts. To support the comparison and development of question answering systems including `SBQA` [7] that is presented further in the text, `SQAD` was supplemented with automatic morphological annotations. The texts were processed with two tools: `Unitok` [12] for text tokenization and `Desamb` [8] morphological tagger, which provides unambiguous morphological annotation of tokenized texts (see Figure 2). Both tools are automatic systems and their accuracy is not 100% thus they occasionally make mistakes. To obtain high-quality data, the tagged texts were checked and corrected by semi-automatic and manual adjustments that are described in the following sections.

3 Adjustments of the `SQAD` Database

In this section, we describe amendments to the tokenization and tagging as obtained by `Unitok` and `Desamb` processing. Some of the modifications were

```
<s>
Kdo      kdo      k3yRnSc1
je       být      k5eAaImIp3nS
autorem  autor    k1gMnSc7
novely   novela   k1gFnSc2
Létající létající k2eAgMnSc1d1
jaguár   jaguár   k1gMnSc1
<g/>
?        ?        kIx.
</s>
```

Fig. 2: Morphological annotation (consists of form, lemma and tag) of a sentence *"Kdo je autorem novely Létající jaguár? (Who is the author of the novel of Flying jaguar?)"*

```
a)  <s>              b)
    1
    200      ⟶           <s>
    300                  1 200 300
    </s>                 </s>
```

Fig. 3: `Unitok`: a) wrong and b) adjusted tokenization of number *"1 200 300"*.

systematic and proceeded in bulk over all the records, some of them were part of manual checking.

3.1 Tokenization Adjustments

During the manual checking of `SQAD` records, we found that large numbers have wrong tokenization if they contained whitespace as a thousands separator (see Figure 3). Therefore we have adapted the matching patterns that `Unitok` uses for processing the text to fix this setup.

3.2 Out-of-Vocabulary Words

The system `Desamb` is used for morphological tags disambiguation according to the word context. In some cases, the context is too narrow or there is no context at all so `Desamb` cannot determine the correct tag. In this case, `Desamb` returns *"k?"* (unknown kind) as the resulting tag. In `SQAD`, this is specifically true for the cases, where the answer in the `SQAD` record contains only a number. Then the `Desamb` resulting tag is *"k?"*. Therefore we have systematically changed such tags to the *"k4"* tag for *numerals* (see Figure 4 for an example).

The `Desamb` tool is working over the attributive Czech tagset of the Majka system [9,10]. `Majka` is based on a deterministic acyclic finite state automaton

```
<s>                           <s>
120   120 k?        ⟶        120   120 k4
</s>                          </s>
```

Fig. 4: `Desamb`: unrecognized number

```
<s>                                     <s>
Los      Los      k?                    Los      Los      k1
Angeles  Angeles  k?                    Angeles  Angeles  k1
</s>                          ⟶         </s>
<s>                                     <s>
LA       LA       k?                    LA       LA       kA
</s>                                    </s>
```

Fig. 5: `Desamb`: unrecognized proper names and abbreviations

(DAFSA) that is created from large morphological database. Despite the coverage of more than 30 million Czech verb forms, `Majka` does not recognize all the existing words, especially proper names and abbreviations. Thus the resulting tag on such words is usually *"k?"*, which means that the system does not contain this word in the database and the context is not long enough to guess the tag. For such words, we have systematically changed the *unknown* tag to:

a) *"k1"* (nouns) for all words that start with an upper case letter, and
b) *"kA"* (abbreviations) for words that contains only upper case letters, words ending with dot or words containing dots between upper case letters.

See Figure 5 for the resulting annotation of unknown proper names and abbreviations.

As we have mentioned above, the `SQAD` database is extracted from Wikipedia texts, which are multilingual and the texts often contain original forms of proper names. For example original writing of the name *"Tokio"* is "東京". These foreign language words are not included in `Majka` database thus `Desamb` always assigns them the tag for unknown words (*"k?"*).

To fix the remaining unknown word tags in `SQAD`, we have extracted all unknown words into one file keeping the original file name, word position and unknown word with its lemma and tag from `Desamb`. This file was than manually annotated and programmatically applied back to the original annotated file (see Figure 6).

3.3 Mistakes in Morphological Analysis

In case of unknown words, the `Desamb` tool can not only leave the word tag undecided, but if the unknown word has a suitable context, `Desamb` can "guess" its lemma and tag even if the word is not present in the `Majka` database. This

Original Desamb output stored in file 03text.txt:

```
Tokio Tokio k1gInSc1
(     (     kIx(
jap.  jap.  kA
東京  東京  k?
```

Record of unknown word extracted from 03text.txt file:

```
./0000/03tex.txt|3|東京   東京   k?
```

Record from 03text.txt with manual changes:

```
./0000/03tex.txt|3|東京   東京   k1
```

File 03text.txt with changes:

```
Tokio Tokio k1gInSc1
(     (     kIx(
jap.  jap.  kA
東京  東京  k1
```

Fig. 6: Desamb: unrecognized foreign words

works very well when Desamb processes Czech words, however it may cause mistakes for foreign words. In this case not only the tag is wrong but also the lemma. For example if we have word "*Las*" (from proper name "*Las Vegas*") the output of Desamb is "Las laso k1gInSc1" (where word *laso* means *lasso*).

To repair all these mistakes, we checked all the SQAD database records and extracted a similar file as in Section 3.2 and corrected the tags in the original SQAD files.

4 Evaluation

We have used the SQAD database to evaluate the accuracy of a first version of SBQA, syntax-based question answering system [7]. SBQA was developed during a master thesis by Matej Pavla at Faculty of Informatics, Masaryk University. The input of SBQA system is a plain text question which is then preprocessed by Unitok and Desamb system and passed to SET [11] parser to identify dependencies and phrase relations within the question. SBQA then decides the rules for finding the answer in its knowledge base based on a match on corresponding syntactic structures (hence the "syntax-base" in its title).

The SBQA knowledge base is made also from plain text documents, which are automatically processed with the same tool chain as the questions (Unitok, Desamb and SET). As we presented in this paper, the automatically preprocessed texts contain mistakes. For the purpose of evaluation of the QA part of SBQA, we have modified its workflow to build the knowledge base from the annotated SQAD database and not to use the automatic processors.

Table 1: Evaluation of SBQA system

total questions	correct	partially correct	incorrect	not found
3,301	758	60	2,003	480
100%	23%	1%	61%	15%

Table 2: Classification of SBQA errors (on 200 examples)

total questions	error in SBQA system	error in tokenization or syntax analysis	uncovered phenomena
200	119	43	38
100%	59.5%	24.5%	19%

For the original results see [7]. The current results of SBQA as measured on the SQAD database are presented in Table 1.

After the evaluation, we have taken 200 questions that are incorrectly answered by SBQA and manually checked the reason of the mistake. We have obtained three categories of mistakes that are caused by:

1. errors in implementation of SBQA system
2. errors in tokenization or syntactic analysis
3. phenomena not covered by the current implementation of SBQA system

An overview of the percentage of these error categories is presented in Table 2. A detailed description of particular error categories can be found in the following subsections.

4.1 Errors in Implementation of SBQA System

The SBQA system balances between answer correctness and the ability to find the answer. Because of that the SBQA implementation uses concept of choosing the answer candidates based on their probability values. On the other hand this implementation sometimes leads to a wrong evaluation that produces incorrect answer.

We have identified the following error types that are caused by SBQA implementation:

- answer in brackets: if the answer of the question is placed in brackets, the SBQA system nearly always provides incorrect answer
- part of speech requirement: even if the tokenization and syntax analysis is provided well, the SBQA system does not check the part of speech that is important for answering the question and the system answers incorrectly
- comparison of dates or numbers: in case of a question, whose answer needs to perform some date or numeric computations, the system does not provide the answer (only exact matches are found).

- wrong question type: as described in [7], SBQA determines for each question its question type. But sometimes the system provides an answer to the question with yes/no question type even if the original question is not a yes/no question.

4.2 Errors in Tokenization or Syntactic Analysis

The SBQA system answers incorrectly in case the tokenization or syntactic analysis contains mistakes.

There are three types of such errors that appear in the current SQAD database:

- Desamb incorrectly detects sentence boundaries and splits one sentence into two or more sentences
- Desamb incorrectly tagged a word thus the syntactic analysis is incorrect and SBQA system cannot derive the required answer
- SET incorrectly parses a sentence and creates an incorrect syntactic tree. This usually leads to incorrect answer.

4.3 Uncovered Phenomena

The SBQA system has not yet implemented advanced NLP techniques such as anaphora resolution. Which means that if the answer to the question is in the sentence that refers to previous text then the answer cannot be found.

5 Conclusions

In this paper we have presented new Czech question answering database called SQAD. Each SQAD record consists of an annotated question, the annotated answer, the annotated sentence containing the full answer, Wikipedia URL as a source of the statement and the author name of this question-answer pair. The morphological annotation was obtained automatically and manually corrected. The corrections make SQAD data more precise so they are more suitable for evaluation of Czech question answering systems.

The SQAD database in its current version is available for download at http://nlp.fi.muni.cz/projects/sqad.

Acknowledgements This work has been partly supported by the Ministry of Education of CR within the LINDAT-Clarin project LM2010013 and by the Czech-Norwegian Research Programme within the HaBiT Project 7F14047.

References

1. Ferrucci, D.A.: IBM's Watson/DeepQA. SIGARCH Comput. Archit. News **39**(3) (2011) – http://doi.acm.org/10.1145/2024723.2019525.

2. Ferrucci, D.A.: IBM's Watson/DeepQA. In: Proceedings of the 38th Annual International Symposium on Computer Architecture. ISCA '11, New York, NY, USA, ACM (2011) – http://dl.acm.org/citation.cfm?id=2000064.2019525.
3. Krishnamurthi, I., Shanthi, P.: Ontology based question answering on closed domain for e-learning applications. Australian Journal of Basic and Applied Sciences **8**(13) (2014) 396–401
4. Chung, H., Song, Y.I., Han, K.S., Yoon, D.S., Lee, J.Y., Rim, H.C., Kim, S.H.: A practical QA system in restricted domains. In: Proceedings of the Workshop Question Answering in Restricted Domains, within ACL. (2004)
5. Vargas-Vera, M., Lytras, M.D.: Aqua: A closed-domain question answering system. Information Systems Management **27**(3) (2010) 217–225
6. Wang, D.: A domain-specific question answering system based on ontology and question templates. In: Software Engineering Artificial Intelligence Networking and Parallel/Distributed Computing (SNPD), 2010 11th ACIS International Conference on, IEEE (2010) 151–156
7. Pavla, M.: Automatic question answering system. Master thesis, Masaryk university, Faculty of Informatics (2014) http://is.muni.cz/th/418138/fi_m/?lang=en.
8. Šmerk, P.: Unsupervised learning of rules for morphological disambiguation. In: Text, Speech and Dialogue, Springer Berlin Heidelberg (2004) http://dx.doi.org/10.1007/978-3-540-30120-2_27.
9. Šmerk, P.: Fast Morphological Analysis of Czech. Proceedings of Recent Advances in Slavonic Natural Language Processing (2009)
10. Jakubícek, M., Kovář, V., Šmerk, P.: Czech morphological tagset revisited. Proceedings of Recent Advances in Slavonic Natural Language Processing **20** (2011) 29–42
11. Kovář, V., Horák, A., Jakubíček, M.: Syntactic analysis using finite patterns: A new parsing system for czech. In: Human Language Technology. Challenges for Computer Science and Linguistics, Berlin/Heidelberg (2011) 161–171 http://dx.doi.org/10.1007/978-3-642-20095-3_15.
12. Michelfeit, J., Pomikálek, J., Suchomel, V.: Text Tokenisation Using unitok. In Horák, A., Rychlý, P., eds.: 8th Workshop on Recent Advances in Slavonic Natural Language Processing, Brno, Tribun EU (2014) 71–75

Semiautomatic Building and Extension of Terminological Thesaurus for Land Surveying Domain

Adam Rambousek, Aleš Horák, Vít Suchomel, and Lucia Kocincová

Natural Language Processing Centre
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
{xrambous,hales,xsuchom2,lucyia}@fi.muni.cz

Abstract. This paper describes the methodology and development of tools for building and presenting a terminological thesaurus closely connected with a new specialized domain corpus. The thesaurus multiplatform application offers detailed information on each term, visualizes term relations, or displays real-life usage examples of the term in the domain-related documents. Moreover, the specialized corpus is used to detect domain specific terms and propose an extension of the thesaurus with new terms. The presented project is aimed at the terminological thesaurus of land surveying domain, however the tools are re-usable for other terminological domains.

Keywords: corpus building, thesaurus, terminological dictionary, term extraction, DEB platform

1 Introduction

Specialists in every field of work use their own domain-specific vocabulary and it is desirable to share the same terminology amongst the professionals. Detailed domain terminology is not usually included in general language dictionaries, thus specialized terminological dictionaries are needed. With the need to share information unambiguously in different languages, terminological dictionaries link original terms to their translations. The taxonomical ordering of the terminology is described by term relations such as synonymy or hyperonymy and hyponymy. The information is presented and visualized in a way that helps the readers (both specialists and general public) to understand the term meaning and usage in contexts. If the data are encoded properly, the system enables automatic processing and integration of the data in third-party applications.

Natural language is still evolving and new words keep appearing or the usage and meaning of words is changing. This evolution is even more noticeable in specialized vocabularies [1]. The thesaurus system thus can employ sophisticated methods of detecting emerging words and distinct new terms in the given domain by processing synchronous domain-oriented corpora.

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2014, pp. 129–137, 2014. © Tribun EU 2014

The Natural Language Processing Centre (NLP Centre) at the Faculty of Informatics, Masaryk University in cooperation with the Czech Office for Surveying, Mapping and Cadastre is developing a system for building and extensions of specialized terminological thesaurus for the domain of land surveying and land cadastre. The project consists of two interconnected parts – an application to create, edit, browse and visualize the terminological thesaurus, and the tools to build large corpus of domain oriented documents with the possibility to detect newly emerging terms, or terms missing from the thesaurus. Already available tools for corpus building and term extraction and the platform for dictionary applications are utilized. During the project, we are enhancing the corpus tools (mainly to support parallel multilingual corpora), building the thesaurus web application (not limited to single domain), and developing methods to inter-connect the domain corpus with the terminological thesaurus.

The project is currently in its first phase. We have built the Czech corpus of land surveying oriented documents and we are able to detect domain specific terms. We have also developed the multiplatform web-based editor and browser thesaurus application based on the dictionary writing platform DEB. Although this project aims to build and manage the terminological thesaurus of land surveying domain, the tools may be re-used for any other domain dictionary, thus stimulating the sharing of information and general awareness of the selected domain.

2 Specialized Corpus and Term Extraction

To build the specialized corpus for land surveying and geoinformation domain, we have followed the principles designed for creation of large corpora extracted and processed from web data. The data for the corpus were gathered from publicly available online resources utilizing two different methods developed by NLP Centre.

Firstly, a set of main websites related to the land surveying, the cadastre of real estates, and related topics was enlisted. See Table 1 for details regarding the sources.

Secondly, based on the content of these *root websites* a broader set of documents from 1,063 websites utilizing the WebBootCat tool [3] was obtained.

Table 1: Website resources for the specialized corpus

Website	Documents	Tokens	Unique documents	Unique tokens
www.cuzk.cz	16,405	3,137,795	15,289	340,943
www.vugtk.cz	4,659	6,419,950	3,212	4,386,238
csgk.fce.vutbr.cz	241	77,255	198	58,561
www.kgk.cz	417	44,814	414	29,890
www.sfdp.cz	192	35,287	106	11,279
www.czechmaps.cz	94	108,506	90	98,914
www.zememeric.cz	8,634	6,100,751	6,200	2,638,308

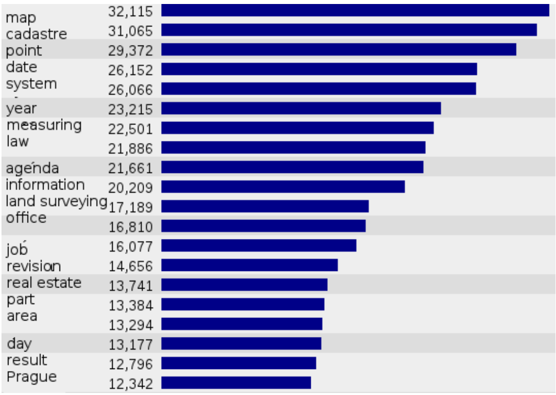


Fig. 1: Most frequent nouns in the land surveying corpora.

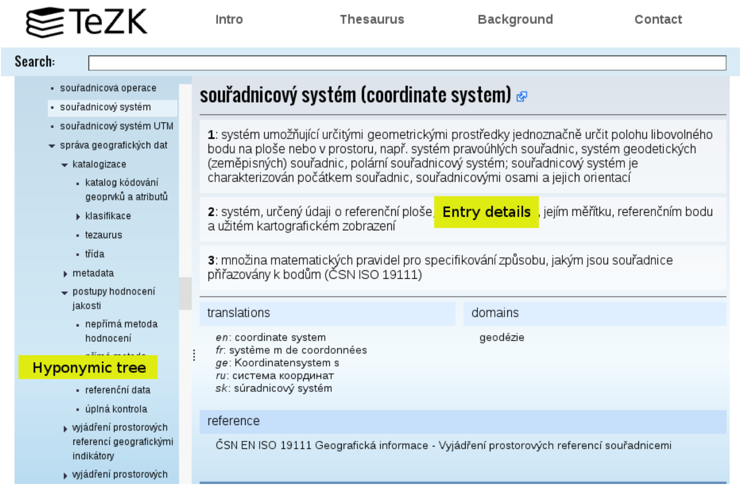
This method needs a set of "seed words" to search the web for relevant documents. We used the main domain terms obtained from existing publicly available terminological dictionary [12] as seed words. The resulting corpus is used for extraction of new suggested terms for inclusion in the thesaurus. See Table 2 for detailed information on downloaded documents and their distribution amongst different sub-domains (as divided in the available terminilogical dictionary) covered by the thesaurus.

Non-textual and low quality content was removed from the downloaded documents, utilizing the Justext tool [2]. Subsequently, duplicate documents or parts (e.g. paragraphs) of the documents were purged with the Onion tool [2].

Following the corpus creation, a list of "candidate terms" (proposals to include into the thesaurus) was prepared. The candidate terms were extracted from the specialized land surveying and geoinformation corpus by employing the process of corpora comparing and keywords extraction [4,5]. Frequencies

Table 2: WebBootCat resources for the specialized corpus

Domain	Documents	Tokens	Unique documents	Unique tokens
GPS system	118	250,833	117	221,315
metrology	144	867,156	144	619,482
photogrammetry	42	244,212	42	227,731
geographical information	55	805,059	55	550,681
mapping	213	858,575	212	722,080
cartography	368	1,358,973	365	1,124,708
cadastre of real estates	260	970,951	259	776,497
geodesy	190	575,381	189	483,679
theory of errors	75	258,345	75	218,809
instrumental technology	115	187,106	113	173,984
engineering surveying	114	286,846	113	242,857



souřadnicový systém (coordinate system)

1: systém umožňující určitými geometrickými prostředky jednoznačně určit polohu libovolného bodu na ploše nebo v prostoru, např. systém pravoúhlých souřadnic, systém geodetických (zeměpisných) souřadnic, polární souřadnicový systém; souřadnicový systém je charakterizován počátkem souřadnic, souřadnicovými osami a jejich orientací

2: systém, určený údaji o referenční ploše, jejím měřítku, referenčním bodu a užitém kartografickém zobrazení

3: množina matematických pravidel pro specifikování způsobu, jakým jsou souřadnice přiřazovány k bodům (ČSN ISO 19111)

translations

en: coordinate system
fr: système m de coordonnées
ge: Koordinatensystem s
ru: система координат
sk: súradnicový systém

domains

geodézie

reference

ČSN EN ISO 19111 Geografická informace - Vyjádření prostorových referencí souřadnicemi

Fig. 2: Browsing the thesaurus, with detailed information for one term.

of words and named-entities in the specialized corpora are compared to the frequencies of the same phrases in a general language corpus (in this case, the biggest Czech corpus developed in NLPC – czTenTen12 [6]). The best candidate terms have the highest frequency quotient [7].

3 DEB Platform

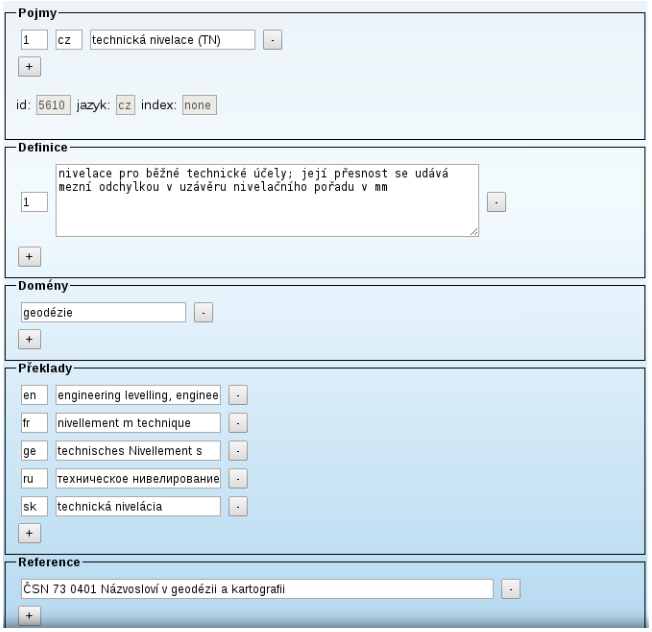
Utilizing the experience from several lexicographic projects, we have designed and implemented universal dictionary writing system that can be exploited in various lexicographic applications to build large lexical databases. The system is called Dictionary Editor and Browser, or DEB [8], and has been used in many lexicographic projects, e.g. for development of the Czech Lexical Database [9], or currently running Pattern Dictionary of English Verbs [10], and Family names in UK [11].

The DEB platform is based on client-server architecture, which brings along a lot of benefits. All the data are stored on a server and considerable part of functionality is also implemented on the server, while the client application can be very lightweight.

This approach provides very good tools for editor team cooperation; data modifications are immediately seen by all the users. Server also provides authentication and authorization tools.

4 Thesaurus Building

Although the main aim of the thesaurus development is publishing the authorized specialized terminology and its updates both to the experts, and general



Pojmy

1 cz technická nivelace (TN)

id: 5610 jazyk: cz index: none

Definice

1 nivelace pro běžné technické účely; její přesnost se udává mezní odchylkou v uzávěru nivelačního pořadu v mm

Domény

geodézie

Překlady

en	engineering levelling, enginee
fr	nivellement m technique
ge	technisches Nivellement s
ru	техническое нивелирование
sk	technická nivelácia

Reference

ČSN 73 0401 Názvosloví v geodézii a kartografii

Fig. 3: Editing the term entry.

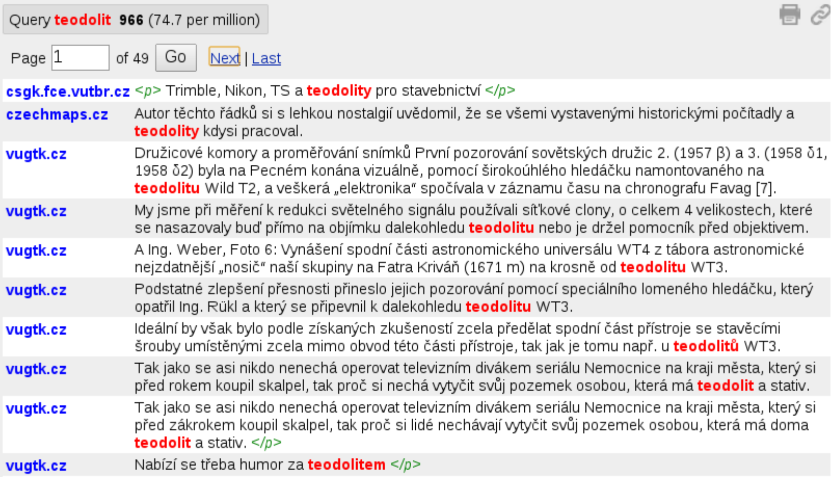
public, the thesaurus will contain broad vocabulary of related terms. Users may search even for unofficial terms and thanks to the relations between the terms and the detailed information on the source of given term, user will find the related terms and links to the recommended official variant.

To build the thesaurus covering broad domain vocabulary, several resources are combined. In the first stage, the current authorized terminological dictionary [12] (containing almost 4,000 terms' definitions and translations, but does not offer the taxonomy network) was combined with the hypero/hyponymic tree of over 6,800 entries (containing hyponymic relations, but no detailed information about terms) and by 450 candidate terms extracted from the domain corpus.

The first two resources were available in HTML form, tagging some parts of entry structure, but still leaving a lot of text in unstructured format. It was necessary to tidy up the data and convert resources to the unified XML format

Table 3: Thesaurus size statistics

total number of terms	8,783	English translations	8,873
hyponymic relations	10,020	German translations	3,936
meaning explanations	4,124	Slovak translations	3,511
		Russian translations	2,762
		French translations	3,936



Query **teodolit** **966** (74.7 per million)

Page 1 of 49 Go Next | Last

csgk.fce.vutbr.cz	<p> Trimble, Nikon, TS a **teodolity** pro stavebnictví </p>
czechmaps.cz	Autor těchto řádků si s lehkou nostalgií uvědomil, že se všemi vystavenými historickými počítadly a **teodolity** kdysi pracoval.
vugtk.cz	Družicové komory a proměřování snímků První pozorování sovětských družic 2. (1957 β) a 3. (1958 δ1, 1958 δ2) byla na Pecném konána vizuálně, pomocí širokoúhlého hledáčku namontovaného na **teodolitu** Wild T2, a veškerá „elektronika“ spočívala v záznamu času na chronografu Favag [7].
vugtk.cz	My jsme při měření k redukci světelného signálu používali síťkové clony, o celkem 4 velikostech, které se nasazovaly buď přímo na objímku dalekohledu **teodolitu** nebo je držel pomocník před objektivem.
vugtk.cz	A Ing. Weber, Foto 6: Vynášení spodní části astronomického universálu WT4 z tábora astronomické nejzdatnější „nosič“ naší skupiny na Fatra Kriváň (1671 m) na krosně od **teodolitu** WT3.
vugtk.cz	Podstatné zlepšení přesnosti přineslo jejich pozorování pomocí speciálního lomeného hledáčku, který opatřil Ing. Rükl a který se připevnil k dalekohledu **teodolitu** WT3.
vugtk.cz	Ideální by však bylo podle získaných zkušeností zcela předělat spodní část přístroje se stavěcími šrouby umístěnými zcela mimo obvod této části přístroje, tak jak je tomu např. u **teodolitů** WT3.
vugtk.cz	Tak jako se asi nikdo nenechá operovat televizním divákem seriálu Nemocnice na kraji města, který si před rokem koupil skalpel, tak proč si nechá vytyčit svůj pozemek osobou, která má **teodolit** a stativ.
vugtk.cz	Tak jako se asi nikdo nenechá operovat televizním divákem seriálu Nemocnice na kraji města, který si před zákrokem koupil skalpel, tak proč si lidé nechávají vytyčit svůj pozemek osobou, která má doma **teodolit** a stativ. </p>
vugtk.cz	Nabízí se třeba humor za **teodolitem** </p>

Fig. 4: Corpus evidence for usage of the selected term (*teodolit*).

for the database storage. Some of the terms were shared by both dictionaries, thus combined term entries were created, containing both detailed information on terms, and the term relations. See Table 3 for more details regarding the current size of the thesaurus.

In the next stage, the thesaurus will be expanded even more by including several resources:

- appropriate parts of the GEMET[1] (General Multilingual Environmental Thesaurus),
- regularly updated Registry of Territorial Identification (RUIAN)[2],
- automatically extracted multi-lingual terms,
- suggestions from the public users.

5 Editing Tool

The thesaurus editing tool is implemented as a client-server application, with DEB server providing the database and management back-end. The client-side application is a multiplatform web application accessible in any modern browser, built utilizing open-source technologies – JQuery[3] and SAPUI5[4] libraries for graphical interface. The client and the server communicate using standardized interface over HTTP, currently JSON format is supported and support for SOAP web-service protocol will be added in the final version.

[1] http://www.eionet.europa.eu/gemet

[2] http://www.cuzk.cz/ruian/

[3] http://jquery.com/

[4] https://sapui5.netweaver.ondemand.com/

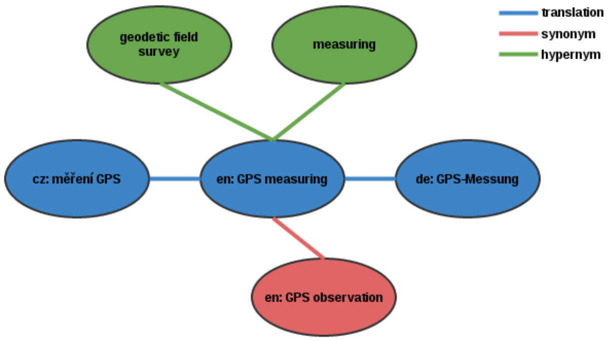


Fig. 5: Entry relations visualized.

The standardized application interface also allows an integration of third-party applications that would like to re-use the thesaurus data. One of the intended use-cases is the integration into the Geoportal[5], where the terms are to be used for the document metadata and categorization.

The thesaurus web application itself provides a graphical interface for browsing the hyponymic tree (see Figure 2). Out of the several possible visualizations of the tree, the expanding multi-level tree was selected, although it may not display all the relations in a proper graph form, it is much more intuitive for the users. If the term has more hyponyms, it is displayed multiple times in the tree structure. To graphically visualize the relations of a term, a graph of hypernyms, synonyms, and other related terms is displayed (see Figure 5).

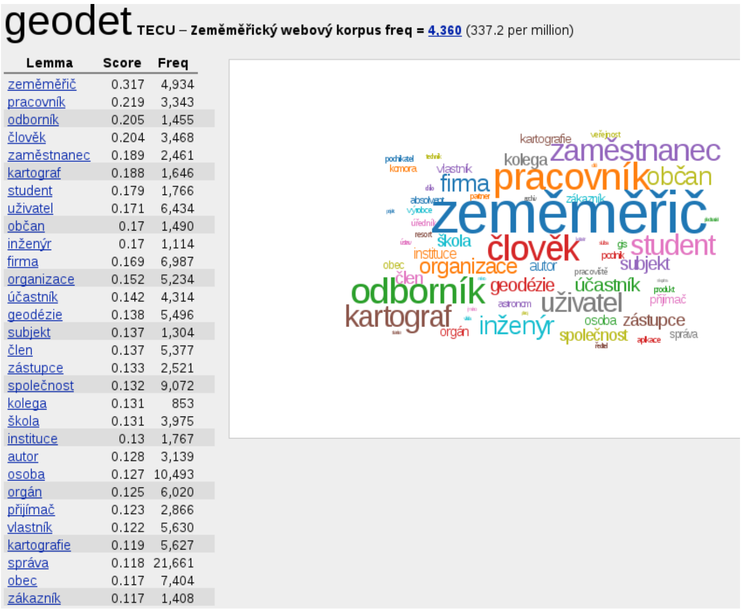
For each term, a detailed description is given, including meaning explanation, translations, or accepted variants. When more sources are incorporated in the thesaurus, the reliability of each source and revision history will be presented to the users. Source reliability follows the rating scale of the Office for Surveying – the most reliable are terms authorized by the terminological committee, followed by terms used in scientific journals, with the terms made up by general public at the bottom of the scale. Users or third-party applications may decide which sources or terms they prefer to work with.

To get a better picture of the term and its usage, extended information from the corpus are presented. Users may consult full examples (see Figure 4) or related words from the corpus (see Figure 6).

6 Conclusions and Future Work

In the next phase of the project, we plan to extend multi-lingual and multi-source aspects of the thesaurus.

[5] `http://geoportal.cuzk.cz/`



geodet TECU – Zeměměřický webový korpus freq = 4,360 (337.2 per million)

Lemma	Score	Freq
zeměměřič	0.317	4,934
pracovník	0.219	3,343
odborník	0.205	1,455
člověk	0.204	3,468
zaměstnanec	0.189	2,461
kartograf	0.188	1,646
student	0.179	1,766
uživatel	0.171	6,434
občan	0.17	1,490
inženýr	0.17	1,114
firma	0.169	6,987
organizace	0.152	5,234
účastník	0.142	4,314
geodézie	0.138	5,496
subjekt	0.137	1,304
člen	0.137	5,377
zástupce	0.133	2,521
společnost	0.132	9,072
kolega	0.131	853
škola	0.131	3,975
instituce	0.13	1,767
autor	0.128	3,139
osoba	0.127	10,493
orgán	0.125	6,020
přijímač	0.123	2,866
vlastník	0.122	5,630
kartografie	0.119	5,627
správa	0.118	21,661
obec	0.117	7,404
zákazník	0.117	1,408

Fig. 6: Related words for the selected term (*geodet*).

Based on the successful evaluation of the Czech specialized corpus for the land surveying domain, we will build corpora in more languages – English, French, German, with the possibility of other languages, depending on the availability of source documents. We will provide automatically extracted terminology from these corpora as the suggestions for terminology translation.

Hand in hand with adding more sources for the thesaurus terms, the editing and browsing application will offer options for filtering the terms based on the source reliability and authorization status and periodic semi-automatic imports from authorized sources.

Acknowledgements This work has been partly supported by TACR in the project TB02CUZK004.

References

1. Fischer, R.: Lexical change in present-day English: A corpus-based study of the motivation, institutionalization, and productivity of creative neologisms. Volume 17. Gunter Narr Verlag (1998)
2. Pomikálek, J.: Removing Boilerplate and Duplicate Content from Web Corpora. Phd thesis, Masarykov university, Faculty of Informatics (2011)
3. Baroni, M., Kilgarriff, A., Pomikálek, J., Rychlý, P.: Webbootcat: instant domain-specific corpora to support human translators. In: Proceedings of EAMT 2006 - 11th Annual Conference of the European Association for Machine Translation, Oslo,

The Norwegian National LOGON Consortium and The Deparments of Computer Science and Linguistics and Nordic Studies at Oslo University (Norway) (2006) 247–252

4. Kilgarriff, A.: Comparing corpora. International journal of corpus linguistics **6**(1) (2001) 97–133
5. Kilgarriff, A.: Simple maths for keywords. In: Proc. Corpus Linguistics. (2009)
6. Suchomel, V.: Recent czech web corpora. In Aleš Horák, P.R., ed.: 6th Workshop on Recent Advances in Slavonic Natural Language Processing, Brno, Tribun EU (2012) 77–83
7. Kilgarriff, A., Jakubíček, M., Kovář, V., Rychlý, P., Suchomel, V.: Finding terms in corpora for many languages with the sketch engine. In: Proceedings of the Demonstrations at the 14th Conferencethe European Chapter of the Association for Computational Linguistics, Gothenburg, Sweden, The Association for Computational Linguistics (2014) 53–56
8. Horák, A., Rambousek, A.: DEB Platform Deployment – Current Applications. In: RASLAN 2007: Recent Advances in Slavonic Natural Language Processing, Brno, Czech Republic, Masaryk University (2007) 3–11
9. Horák, A., Rambousek, A.: PRALED – A New Kind of Lexicographic Workstation. In: Computational Linguistics. Springer (2013) 131–141
10. Hanks, P.: Corpus pattern analysis. In: Proceedings of the Eleventh EURALEX International Congress, Lorient, France, Universite de Bretagne-Sud (2004)
11. Hanks, P., Cullen, P., Draper, S., Coates, R.: Family names of the United Kingdom. (2014)
12. Hánek, P.: Terminologický slovník zeměměřictví a katastru nemovitostí. Výzkumný ústav geodetický, topografický a kartografický, v.v.i. (2012)

Mapping Czech and English Valency Lexicons: Preliminary Report

Vít Baisa, Karel Pala, Zdeňka Sitová, and Jakub Vonšovský

Natural Language Processing Centre
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
{xbaisa,pala,xsitova,xvonsovs}@fi.muni.cz

Abstract. We describe here a very first attempt to connect two valency lexicons: Pattern Dictionary of English Verbs (PDEV) and VerbaLex. Both lexicons contain verbs together with their syntactic structure (arguments of the verbal predicate) and semantic restrictions (semantic types typical for a given verb argument). The lexicons are similar in overall but differ in details since their formalisms are tailored for the respective languages. They also differ in a way they have been built: whilst the former resource has been built using Corpus Pattern Analysis methodology the latter has been built upon previous datasets Brief, Vallex and Czech WordNet. We present a preliminary work on linking English patterns in PDEV with their Czech equivalents: frames in VerbaLex.

Keywords: valency, lexicon, PDEV, CPA, VerbaLex, ontology, WordNet

1 Introduction

Valency lexicons are lexical resources containing valency frames (patterns) of individual verbs. The frames contain information about verb arguments (such as direct and indirect object, subject), their morphosyntactic properties (such as cases in Czech) and their semantic roles (such as agens, patient, instrument).

For Czech there are two valency lexicons, one is Vallex [1] by Žabokrtský (approximately 6,000 Czech verbs) based on formalism of Functional Generative Description (FGD) and VerbaLex [2] developed by Hlaváčková et al. (approximately 10,500 Czech verbs). For English we use PDEV by Hanks et al. [3].

The mentioned valency lexicons for Czech basically share the morphosyntactic information (about cases and adverbial phrases) but they differ in their inventories of the semantic roles: Vallex uses about 40 roles, VerbaLex uses complex roles consisting of the main roles (48) and selectional restrictions (900).

In the PDEV, the description of the morphosyntactic properties of the verb arguments is different from the Czech lexicons as English displays the fixed word order (SVOMPT). The semantic roles and types are based on Pustejovsky's shallow ontology containing 228 items.

Our goal is to exploit the overall similarity (structure of frames and patterns) and propose possible equivalents of English patterns and Czech frames. In this paper we present a preliminary analysis of correspondences between the two lexical resources. We believe that the resulting translation valency dictionary would be very useful resource for natural language processing tasks, mainly for machine translation.

2 Pattern Dictionary of English Verbs

The PDEV[1] [4] is a result of a long-term work by Patrick Hanks and his colleagues. Currently, it is being developed within project *Disambiguation of Verbs by Collocation*[2] (DVC) at University of Wolverhampton.

The method of building the lexicon is based on finding corpus evidence: the English verb patterns are created only when observed in a sample of corpus examples for a given English verb. This technique of *Corpus Pattern Analysis* (CPA) was invented by Patrick Hanks [4]. The corpus used in CPA is the written part of British National Corpus.

The focus of CPA is on the prototypical syntagmatic patterns with which verbs in use are associated. Verb patterns in PDEV consist not only of the basic *argument structure* or *valency structure* of each verb (typically with semantic values stated for each of the elements), but also of subvalency features, where relevant, such as the presence or absence of a determiner in noun phrases constituting a direct object. For example, the meaning of *take place* is quite different from the meaning of *take his place*. The possessive determiner makes all the difference to the meaning in this case.

3 VerbaLex

The Czech lexicon[3] [2] has been initially based on the following resources:

1. the starting repertoire of the verbs has been taken from syntactic lexicon of verb valencies called BRIEF by Pala and Ševeček [5],
2. Czech WordNet valency lexicon developed within the Balkanet project.
3. The tool for handling the structure of the lexicon has been partially inspired by the editor developed for the above-mentioned Vallex. A new editor has been developed and is used for editing and browsing VerbaLex.

The verbs in VerbaLex are grouped into synsets in the same way as in Princeton WordNet [6]. Approximately 8,000 of them are linked to the equivalent English WordNet synsets.

[1] `http://www.pdev.org.uk`

[2] `http://clg.wlv.ac.uk/projects/DVC/`

[3] `http://nlp.fi.muni.cz/verbalex/html2/generated/alphabet/`

4 Related work

We have already mentioned Vallex as a similar resource for Czech. Framenet for English [7] should be mentioned as well though it is not only a verb lexicon. Valency lexicons are available for a number of languages: German [8], French, Italian, Russian [9], Polish [10] and others.

There have been some attempts at linking valency lexicons, e.g. [11] describes their ongoing efforts in aligning two valency lexicons PDT-VALLEX and EngValLex on the basis of a parallel treebank. The token alignment is done manually by annotators whose task is to go through the verb occurrences in the treebank, collect a typical representative of a frame mapping and control and decide potential conflicting cases. Once collected, the frame mapping is automatically applied to all its other potential representantives.

Related to our effort is also EngValLex [12]—transformation of the PropBank [13] lexicon to the structure of Vallex. After linguistic comparison of PropBank and Vallex, PropBank was automatically converted to FGD-compliant form which was later manually refined. The method is as follows: first, all slots have been renamed using functors, second, the non-obligatory free modifiers have been deleted and optional elements marked. Third, frames corresponding to the same verb sense have been merged. Fourth, the lexicon has been refined in the process of treebank annotation by addition of other frames, whole verb lemmas, and also, the PropBank adapted frames were corrected manually with respect to the language data available in the English part of parallel treebank. [11]

5 Analysis of differences and similarities

For this study, the verbs have been selected in the way that there was only one pattern in PDEV which helps the translation into VerbaLex and avoids ambiguity. There are 313 single-pattern verbs in PDEV. For some of them it is not possible to find Czech translation equivalents,[4] thus they have been left out from further analysis.

There are some features (grammatical categories) in Czech that do not have their respective counterparts in English. One of them is category of aspect: in the regular cases in which the members of an aspect pair preserve the same meaning, the category of aspect can remain in the frames, as in pair like *zrychlit, zrychlovat* (to accelerate). Similar category that should be preserved is category of case (7 grammatical cases in Czech).

Some verbs in PDEV which simply do not have direct translation equivalents in VerbaLex (*calcify, demystify, ignore, . . .*) are excluded from further considerations. On the other hand, there are many verbs in VerbaLex for which we cannot find the translational equivalents in PDEV because it is too small so far[5]

[4] This is caused by special terminology from very limited domains in BNC.

[5] There are roughly 1,100 completed verbs in PDEV.

or because many complex Czech verbs can not be translated on lexical level, for example: *povytáhnout* (to pull something out a bit), *poposednout si* (to move on a bit) etc.

If we look at PDEV and VerbaLex we can observe that their ontological structures are considerably different which complicates the mapping. The ontology in VerbaLex is partly based on the Top Ontology used in EuroWordNet [14] and on selected literals from Princeton WordNet. In PDEV *Shallow Ontology* by Pustejovsky [15] is used. For example, *Group* class in PDEV contains subclasses *Human Group, Vehicle Group, Animal Group, Physical Object Group* which have their own respective categories in VerbaLex. Only very few classes inherit their mapping such as PDEV *Machine* → `<artifact:1>` in VerbaLex. This means we have to uncover relations between every single class by analysing more and more words. Nevertheless, so far it seems we can go up in the classes to find a match such as for *water* which corresponds to `SUBS<liquid substance:1>` in VerbaLex and has its own class in PDEV. In one of our analyses it maps `SUBS<liquid substance:1>` to *Entity* having *Water* class as one of its descendants.

From 21 analyses, 9 patterns were mapped without any problems from one lexicon to another. 10 patterns were mapped with some imperfections such as missing frames in PDEV (for example out of 5 frames in Verbalex only 2 had a match in PDEV) or small mismatches in frames (obligatory requirement in Verbalex). Those small mismatches in frames which happened in 2 cases could be somehow penalized in an automatic tool. Only one record was unmappable (*burrow*) because frames were mismatched (Verbalex did not cover case of burrowing animals) and one record was not present at all in Verbalex (*disregard*). For some examples, see Table 1.

Table 1: Some mapping examples, PDEV on left, VerbaLex on right

<table>
<tr><th><b>Animate physical object</b></th><th></th></tr>
<tr><td><i>Human Group | Human</i></td><td>AG<person|child|...></td></tr>
<tr><td><i>Animal | Bird</i> (all animals)</td><td>AG<animal></td></tr>
<tr><td><i>Institution</i></td><td>GROUP<institution>|AG<person></td></tr>
<tr><td><b>Precise mappings</b></td><td></td></tr>
<tr><td><i>Machine</i></td><td>ART<artifact>|INS<device></td></tr>
<tr><td><i>Body Part</i></td><td>PART</td></tr>
<tr><td><i>Artwork</i></td><td>COM<written communication></td></tr>
<tr><td><i>Fluid | Beverage</i></td><td>SUBS<liquid substance></td></tr>
<tr><td><b>Imprecise mappings</b></td><td>(one of possible mappings)</td></tr>
<tr><td><i>Action</i></td><td>MAN,how</td></tr>
<tr><td><i>Activity</i></td><td>ACT<act></td></tr>
<tr><td><i>Eventuality</i></td><td>Event</td></tr>
<tr><td><i>Entity</i></td><td>GROUP<institution></td></tr>
<tr><td><i>Physical Object</i></td><td>OBJ</td></tr>
<tr><td><i>Anything</i> causes <i>Anything</i></td><td>REAS<reason>,díky,kvůli</td></tr>
</table>

In the mapping, `AG` (agens) can also be `PAT` (patient), `ENT` (entity) or `SOC` (associate) thus *Human* would map to `AG<person>` as well as to `PAT<person>`.

Record analysis example

VerbaLex frames (VN) and PDEV patterns (PN) are in typewriter typeface. PDEV pattern implicatures are in italics.

rozmazlovat (cosset)
[V1] `AG <person:1> V PAT<person:1>`
[V2] `AG <person:1> V PAT<person:1> (ACT<act:2>)`
[P1] `[[Human 1]] cosset [[Human 2]]`
[[Human 1]] cares for [[Human 2]] in an excessively protective and fussy way
Comment: exact match.

zakrýt (blanket)
[V1] (překrýt) `OBJ <object> V OBJ<object>`
[V2] (překrýt) `OBJ <object> V OBJ<object> PART<part>`
[V3] `AG<person> V PAT<person> (ART<covering>)`
[P1] `[[Stuff|{PhysicalObject1=PLURAL}]] blanket [[Location|PhysicalObject2]]`
[[Location|Physical Object 2]] becomes covered by a layer of
[[Stuff|Physical Object 1 = PLURAL]]
Comment: *Physical Object* is mapped to `OBJ<object>` here, but in fact it is still quite general class.

zbankrotovat (bankrupt)
[V1,2] `ENT<person|instit> V (REAS<reason>díky,kvůli)`
[P1] `[[Human1|Instit1|Event]] bankrupt [[Human2|Instit2]]`
[[Human1|Instit1|Event]] causes [[Human2|Instit2]] to not have enough money to pay his or her or its debts
Comment: *Human 2 | Institution 2* → to `<person>|<institution>`. `REAS<reason>` maps to *A* causes *B*.

6 Discussion & conclusions

So far we have uncovered several promising relations between the two lexicons. Unfortunately, as their ontology structures are completely different, we would need to analyse tens or hundreds possible pairs to get a more complex image of possible mappings. So far from about 20 records we are already able to map animate physical objects and some of inanimate objects which together form the biggest group in PDEV. This is a basis for further investigation and for a rule-based approach to proposing and linking possible equivalents from PDEV and VerbaLex. The main problems consist of the following:

1. the ontologies used in PDEV and VerbaLex are structured differently, there is a shallow ontology in PDEV and a sort of the Aristotelian ontology based on Top Ontology from EuroWordNet in VerbaLex.
2. Basic items in VerbaLex are synsets containing usually more than one verb lemma, whereas in PDEV the basic items are the individual verb lemmas. This, however, can be handled by obtaining appropriate lists from VerbaLex (by expanding and filtering the verb list).

The comparison of the two ontologies is a separate task that should be further investigated and deserves a separate paper.

We hope that in the near future we will be able to propose and implement an automatic tool with high accuracy of PDEV patterns translations of VerbaLex frames and vice versa.

Acknowledgement This work was partially supported by the Ministry of Education of CR within the LINDAT-Clarin project LM2010013 and by the Czech-Norwegian Research Programme within the HaBiT Project 7F14047.

References

1. Žabokrtský, Z., Lopatková, M.: Valency information in vallex 2.0. The Prague Bulletin of Mathematical Linguistics (87) (2007) 41–60
2. Hlavácková, D., Horák, A.: Verbalex–new comprehensive lexicon of verb valencies for czech. In: Proceedings of the Slovko Conference, Citeseer (2005)
3. Hanks, P., Pustejovsky, J.: A pattern dictionary for natural language processing. Revue française de linguistique appliquée **10**(2) (2005) 63–82
4. Hanks, P.: Lexical Analysis: Norms and Exploitations. Mit Press (2013)
5. Pala, K., Ševeček, P., et al.: Valence českỳch sloves. (1997)
6. Fellbaum, C.: WordNet. Wiley Online Library (1998)
7. Baker, C.F., Fillmore, C.J., Lowe, J.B.: The berkeley framenet project. In: Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1, Association for Computational Linguistics (1998) 86–90
8. Hinrichs, E.W., Telljohann, H.: Constructing a valence lexicon for a treebank of german. In: Proceedings of the Seventh International Workshop on Treebanks and Linguistic Theories. (2009) 41–52
9. Lyashevskaya, O.: Bank of russian constructions and valencies. In: LREC. (2010)
10. Przepiórkowski, A.: Towards the design of a syntactico-semantic lexicon for polish. In: Intelligent Information Processing and Web Mining. Springer (2004) 237–246
11. Šindlerová, J., Bojar, O.: Building a bilingual vallex using treebank token alignment: First observations. In: Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10). (2010)
12. Cinková, S.: From propbank to engvallex: Adapting the propbank-lexicon to the valency theory of the functional generative description. In: Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'06). (2006)
13. Palmer, M., Gildea, D., Kingsbury, P.: The proposition bank: An annotated corpus of semantic roles. Computational Linguistics **31**(1) (2005) 71–106

14. Vossen, P.: A multilingual database with lexical semantic networks. Springer (1998)
15. Rumshisky, A., Hanks, P., Havasi, C., Pustejovsky, J.: Constructing a corpus-based ontology using model bias. In: FLAIRS Conference. (2006) 327–332

Tools for Fast Morphological Analysis Based on Finite State Automata

Pavel Šmerk

Natural Language Processing Centre
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
smerk@fi.muni.cz

Abstract.
The paper presents a new implementation of some of Jan Daciuk's algorithms and tools for morphological analysis based on finite state automata [1]. In particular, we offer a reimplemented version of the tool which builds the automata from an input set of strings and of the tool which performs the morphological analysis itself. In addition to 8-bit versions we also offer "Unicode-aware" versions with the Unicode characters encoded directly in the arcs of the automaton. The new implementation is faster than the original one and its code is much more simple and straightforward.

Keywords: morphological analysis, minimal deterministic finite state automata

1 Introduction

Computational morphological analysis is one of the first steps in the automatic treatment of natural language texts. Just after splitting the processed text into words we usually need a tool which for each such word returns its possible corresponding lexical entries (*lemmata*) and a relevant grammatical information. For languages with limited compounding and with morphology realized mainly by changes at the end of the word (Slavic languages can be taken as an example), it is possible to describe their morphology by means of a simple list of all words (word forms) and their possible interpretations and to let the morphological analyzer only search this list for each input word.

Of course, it would not be feasible to search such a list directly due to its huge size. However, the list can be viewed as a finite (formal) language and consequently it can be represented by a minimal deterministic acyclic finite state automaton, which can be pretty small. The morphological analyzer then only follows a path in the automaton according to letters of the analyzed word and, if a corresponding path exists, returns all possible continuations of this path as a result.

In the following section we describe the input data format and illustrate how the morphological analysis works. In the next section we present results

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2014, pp. 147–150, 2014. © Tribun EU 2014

of newly reimplemented tools and finally we discuss some possible future modifications.

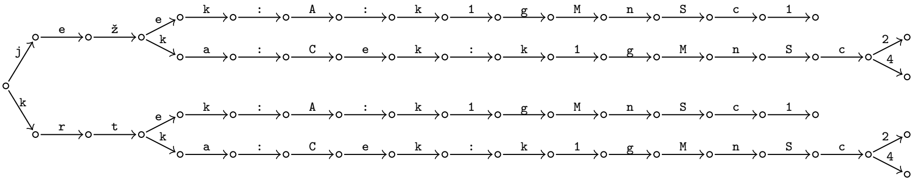
2 Data for morphological analysis

The data for morphological analysis are simply a list of all combinations of recognized input strings and corresponding outputs of the analyzer, where pairs of two words are encoded as pairs formed by the first word and a difference between the words [2]. For example, in the following part of data for word form → lemma + tag analysis (with the original data on the right side and the encoded form on the left side)

```
ježek:A:k1gMnSc1       ←  ježek:ježek:k1gMnSc1
ježka:Cek:k1gMnSc2     ←  ježka:ježek:k1gMnSc2
ježka:Cek:k1gMnSc4     ←  ježka:ježek:k1gMnSc4
krtek:A:k1gMnSc1       ←  krtek:krtek:k1gMnSc1
krtka:Cek:k1gMnSc2     ←  krtka:krtek:k1gMnSc2
krtka:Cek:k1gMnSc4     ←  krtka:krtek:k1gMnSc4
```

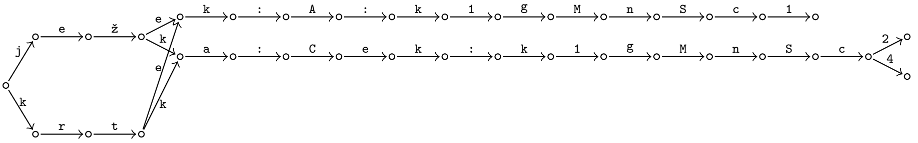
the (first) colon is a delimiter between the possible inputs and corresponding outputs and the letters A and C as the first and the third letters of the alphabet mean "to get the lemma delete n-1 (i.e. 0 or 2, respectively) last characters from the word form and then attach the rest of the string (i.e. empty string or ek, respectively)". For example, the word form krtek will be analyzed as a lemma krtek with a morphological tag k1gMnSc1[1] and a word form krtka as a lemma krtek with morphological tags k1gMnSc2 or k1gMnSc4[2].

Such a list is then represented as a minimal deterministic acyclic finite state automaton using Jan Daciuk's algorithms for incremental building of minimal DAFSAs [1]. The following graph corresponds to the non-minimized automaton (trie)



j e ž
e k : A : k 1 g M n S c 1
k a : C e k : k 1 g M n S c 2 4
k r t
e k : A : k 1 g M n S c 1
k a : C e k : k 1 g M n S c 2 4

and the second graph corresponds to the minimized automaton used for the morphological analysis.



[1] *mole* in nominative form

[2] *mole* in genitive and accusative form

This representation dramatically reduces the size of the data (some particular figures can be seen later in Table 1). The lookup is then very simple: if the analysed string concatenated with the delimiter is found in the automaton, then each possible remaining path to a final state of the automaton encodes one of possible analyses.

It means that there is no "real" analysis as any sophisticated algorithm above some grammar model or a system of paradigms, but whole analysis is only a simple — and therefore fast — dictionary search.

3 Experiments and results

3.1 Building the data for morphological analysis

We demonstrate our results on four sets of data. English and Russian morphological data are from the project FreeLing, the data for Czech are ours. We compare our new implementation with the original Daciuk's implementations[3] and with Java reimplementation of David Weiss[4] from project Morfologik (it also offers a more compact format at a price of a greater build time, for details refer to [3]). For the purpose of comparison, our implementation produces binary identical output (except for custom header) as the original Daciuk's fsa_build built with `-DFLEXIBLE -DNEXTBIT -DSTOPBIT` compile options. The first table describes the data sets and in the last column is the size of the resulting automaton (both input and output sizes are in bytes).

Table 1: Data sets used in the experiments.

data set	input size	words (lines)	output size
EN	1,417,920	88,652	244,764
RU	114,605,988	2,844,516	3,639,960
CZ free	105,001,670	3,393,080	931,594
CZ full	828,973,970	27,764,093	3,795,423

The second table presents build times for the three compared tools.

Table 2: Build times in seconds.

data set	fsa_build	morfologik	new implem.
EN	0.24	0.59	0.08
CZ free	12.63	7.50	4.19
RU	26.04	10.19	9.41
CZ full	121.41	57.21	41.71

[3] `www.eti.pg.gda.pl/katedry/kiw/pracownicy/Jan.Daciuk/personal/fsa.html`, version 0.51

[4] `http://sourceforge.net/projects/morfologik/`, version 1.9.0

3.2 Morphological analyzer and UTF-8 versions

We also offer a new implementation of morphological analyzer. Unfortunately, we were not able to make fsa_morph case conversions work even with Daciuk's original language files, thus it is difficult to compare analysis times in real world scenarios. If we changed our analyser to immitate (somewhat broken) fsa_morph output for 10 million Czech words from corpus, we are ca. 20 % faster (12.51 s × 15.25 s).

Daciuk's tools allow to be compiled with UTF-8 support, but it requires the user to describe the case conversions and diacritics adding/removal. We have UTF-8 variants of our tools which works with automata with UTF-8 labels and our case conversions and diacritics restoration follows the Unicode standard, which means one solution for all languages.

Except for speed, another advantage of our solution is much shorter and straighforward code. It is not easy to make a fair comparison, but, for example, files needed for fsa_build have more than 200 kB in total, whereas the code of our new implementation has less than 20 kB. It is obvious that in case of such a huge difference it is easier to maintain, adjust and further develop the shorter code.

The new tools are accessible from `http://nlp.fi.muni.cz/ma`.

4 Future work

The new tools are ready to use and the presented results are promising, but it is still a work in progress. We plan to further reduce both time and final size of the automata construction. We want to employ some variable length encoding of unicode codepoints, numbers and addresses (similar to [1], but computationally simpler one). We suspect Daciuk's "tree index" used to discovering already known nodes during the automaton construction to be slow for large data and we hope that simple hash will decrease the compilation time significantly.

Acknowledgements This work has been partly supported by the Ministry of Education of CR within the Lindat Clarin Center LM2010013.

References

1. Daciuk, J.: Incremental Construction of Finite-State Automata and Transducers, and their Use in the Natural Language Processing. PhD thesis, Technical University of Gdańsk, Gdańsk (1998)
2. Kowaltowski, T., Lucchesi, C.L., Stolfi, J.: Finite Automata and Efficient Lexicon Implementation (1998) Technical Report IC-98-02, University of Campinas, São Paulo.
3. Daciuk, J., Weiss, D.: Smaller representation of finite state automata. Theoretical Computer Science **450**(0) (2012) 10–21

Subject Index

Author Index

RASLAN 2014

Eighth Workshop on Recent Advances in Slavonic Natural
Language Processing

Editors: Aleš Horák, Pavel Rychlý
Typesetting: Adam Rambousek
Cover design: Petr Sojka

Printed and published by Tribun EU s. r. o.
Cejl 32, 602 00 Brno, Czech Republic

First edition at Tribun EU
Brno 2014

ISSN 2336-4289

www.librix.eu