RASLAN 2016
Recent Advances in Slavonic Natural Language Processing

A. Horák, P. Rychlý, A. Rambousek (Eds.)

RASLAN 2016

Recent Advances in Slavonic Natural Language Processing

Tenth Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2016
Karlova Studánka, Czech Republic,
December 2–4, 2016
Proceedings

Tribun EU
2016

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`

Adam Rambousek
Faculty of Informatics, Masaryk University
Department of Information Technologies
Botanická 68a
CZ-602 00 Brno, Czech Republic
Email: `rambousek@fi.muni.cz`

ISBN 978-80-263-1095-2
ISSN 2336-4289

Preface

This volume contains the Proceedings of the Tenth Workshop on Recent Advances in Slavonic Natural Language Processing (RASLAN 2016) held on December 2nd–4th 2016 in Karlova Studánka, Sporthotel Kurzovní, Jeseníky, Czech Republic.

The RASLAN Workshop is an event dedicated to the exchange of information between research teams 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 verified methods together with descriptions of development trends. The workshop also serves as a place for discussion 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 parsing
* 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 2016 offers a rich program of presentations, short talks, technical papers and mainly discussions. A total of 18 papers were accepted, contributed altogether by 28 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 Marie Stará. 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 a printer of these proceedings is gratefully acknowledged.

Brno, December 2016 Karel Pala

Table of Contents

Part I

Text Corpora

Between Comparable and Parallel: English-Czech Corpus from Wikipedia

Adéla Štromajerová, Vít Baisa, Marek Blahuš

Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
`{xstromaj,xbaisa,xblah}@fi.muni.cz`

Abstract. We describe the process of creating a parallel corpus from Czech and English Wikipedias using methods which are language independent. The corpus consists of Czech and English Wikipedia articles, the Czech ones being translations of the English ones, is aligned on sentence level and is accessible in Sketch Engine corpus manager.[1]

Key words: parallel corpora, comparable corpora, Wikipedia

1 Introduction

Wikipedia is now available in almost 300 languages, 13 of them having more than one million articles. The largest is the English Wikipedia, which contains more than 5 million articles. For each article, Wikipedia stores information about all the editing: the editor, the time of the editing and the changes made in the article. It is also possible to view any previous version of the article.

New articles are either written from scratch, or an article from another language version of Wikipedia is translated. However, translations do not need to cover the whole original article. Translations in Wikipedia are mostly from English to other languages. The *Translated page* template should be always added into such article so that it is clear that it is a translation and to identify what article and what language version of Wikipedia have been used for the translation.[2]

We used this information and extracted the translated articles as a basis for an English-Czech parallel corpus. There are more than 37,000 such articles[3] and they cover a wide range of topics. As the whole Czech Wikipedia contains about 350,000 articles, the proportion of the English-Czech translations is quite high, i.e. approximately 10%, when the number of articles (not their length) is considered.

[1] `https://ske.fi.muni.cz`
[2] `https://cs.wikipedia.org/wiki/Wikipedie:WikiProjekt_P%C5%99eklad/Rady`
[3] `https://cs.wikipedia.org/wiki/Kategorie:Monitoring:%C4%8Cl%C3%A1nky_p%C5%99elo%C5%BEen%C3%A9_z_enwiki`

2 Related work

There have been a few attempts at creating corpora from Wikipedia, e.g. [1]. There is a huge comparable corpus *Wikipedia Comparable Corpora*[4]. This consists of monolingual corpora, every one of them containing all articles from a particular language version of Wikipedia. These corpora are then document-aligned, i.e., the corpora consist of document pairs of articles on the same subject. There are also some parallel corpora based on Wikipedia, for example, a Chinese-Japanese parallel corpus created to help to improve the SMT between these two languages [2].

The field of web parallel corpora is of great interest nowadays, and there are many projects dealing specifically with Wikipedia (Besides the already mentioned ones, e.g. a Persian-English parallel corpus [3]. However, there is no parallel corpus made out of English and Czech articles from Wikipedia.

3 Exploiting Wikipedia Article Translations

The workflow was the following: a) to identify which Czech articles were created by translating English articles; b) to find out which version of the Czech article was the first, original translation; c) to identify the English articles from which the Czech ones were translated; d) to determine the version of the English article from which the Czech one was translated; and e) to download the texts of the Czech articles and the corresponding texts of the English articles.

First, it was necessary to determine which Czech articles were translated from English. This was quite simple as it is required to include the Translated page template in all the translated pages in Wikipedia. This template is supposed to be inserted into the References section and the structure of the Czech template, called Šablona:Překlad, is as follows: `{{Překlad|jazyk= |článek= |revize= }}` or, in short, `{{Překlad|en|article|123456}}`, where the second field denotes the language of the original article (represented by a language code, e.g., en for English), the third gives the name of the original article and the last field stands for the version identifier of the revision of the original article from which the Czech one was translated. The version identifier is a number and can be found at the total end of the permanent link to a given article/version, preceded by `oldid=`.

Second, it had to be identified which version of the Czech article was the first, original translation. If the current version of the Czech article was downloaded, the English and Czech texts could differ substantially as the articles change over time and some parts of the original translation would be edited or deleted, while some others would be added. Therefore, it would be then a more difficult task for a sentence aligner to extract parallel sentences from such different texts. Therefore, the revisions of Czech articles had to be searched and the revision where the full Translated page template appeared for the first time, had to be downloaded.

[4] `http://linguatools.org/tools/corpora/wikipedia-comparable-corpora/`

For accessing its data, meta-data and features, Wikipedia provides users with various APIs connected to MediaWiki. MediaWiki[5] is a free software open source wiki package written in PHP, which was originally designed for Wikipedia, but nowadays it is used by many other wikis. Its most known API is the *MediaWiki action API*. This provides a direct access to the data in MediaWiki databases via a URL. Clients then request particular `action` parameter to get the desired information.[6] Using `action=query` module, it is possible to get meta information about the wiki, properties of pages, or lists of pages matching certain criteria[7]. IDs of all the articles translated from the English Wikipedia were retrieved with the help of this module.

We decided to retrieve Wikipedia pages in HTML format. The process of downloading a language pair of a Czech and an English article was the following. Taking the page ID from the ID list retrieved before, the revisions of the particular Czech article were accessed. They were listed in the order from the oldest to the newest. For every revision, its ID, timestamp and content were listed. The content of them was then searched for the *Translated page* template using regular expressions. The target of the search was only the full template containing not only the language and the name of the article, but also the revision ID. There were some articles which contained only an incomplete template without the revision ID. These articles were not downloaded as it was not possible to determine exactly from which version the Czech translation was made[8]. The ID of the identified revision was then taken and this particular version of the Czech article was retrieved.

We then used *jusText*[9] [4]. It removes all the boilerplate content, such as navigation links, headers and footers, and it preserves the text in the form of a list of paragraphs.

Another step was to remove the final sections of Wikipedia articles which were needless in the text, i.e., References, External links, etc. Finally, other unnecessary parts were removed from the page, e.g., reference numbers and note numbers. After this, the title and then the rest of the Czech text was written into a document with a name consisting of a number followed by the code for the Czech language, i.e., "cs".

It has to be noted that we could work with MediaWiki format of articles, but there is no suitable method of converting MediaWiki format into HTML reliably in a large scale.

The English pages were then processed in a similar way.

[5] `https://www.mediawiki.org/wiki/MediaWiki`

[6] `https://www.mediawiki.org/wiki/API:Main_page`

[7] `https://www.mediawiki.org/wiki/API:Query`

[8] The amount of such articles was, however, very low, and together with other download errors caused by inconsistencies in metadata, etc., it was lower than 1% of the total sum of 37,00 articles for both Czech and English.

[9] `http://corpus.tools/wiki/Justext`

4 Parallel Sentences Extraction

The retrieved texts cannot be considered parallel. Therefore, they had to be processed before including them in the corpus. With the help of a sentence aligner, sentences in the corresponding texts were aligned. These aligned sentences can then be considered parallel and can be used as data for a parallel corpus.

There are many sentence aligners available. We used *hunalign* [5] because it is easy to use, it is quick, it is a free software, and it supports not only one-to-one alignments, but also m:n mapping.

Sentence aligners use mostly three types of alignment methods: length-based, dictionary or translation-based and partial similarity-based. Hunalign is a hybrid sentence aligner whose alignment method is length-based and dictionary-based. These methods are simpler than, e.g., the translation-based method. Sentence aligners using the translation-based method, such as Bleualign [6], first make a machine translation of the source text and then compare this translation to the target text. Because of this machine translation included, translation-based alignment methods are quite complex and time-consuming. There are also sentence aligners that first need big corpora to be trained on (e.g., Gargantua [7]), which, again, makes their use more demanding both of resources and time.

Hunalign combines the length-based and the dictionary-based methods and aligns the sentence segments according to scores from both methods together. If there is no dictionary provided, it first aligns according the sentence length, and then, based on this alignment, makes an automatic dictionary and realigns the texts according to this dictionary.

As hunalign does not come with an English-Czech dictionary, the next step was to get such a dictionary in the format suitable for hunalign. This dictionary was created from monolingual dictionaries provided with *LF Aligner*. LF Aligner is a hunalign wrapper written by Andras Farkas[10]. It comes with built-in monolingual dictionaries which are then during the process of aligning combined into bilingual dictionaries according to the languages used. The English and the Czech dictionary were taken and they were combined into an English-Czech dictionary according to the structure of hunalign dictionaries, which is `target_language_phrase @ source_language_phrase` per line. The resulting dictionary was then provided to hunalign to achieve a better alignment.

During the alignment process, it was found out that hunalign is not able to produce alignment of files which differ in sizes considerably. In this case, the alignment is not produced. However, such files are unalignable in general, regardless of the used sentence aligner. The amount of files not aligned was quite big, i.e., about 20%. However, the remaining data were still enough for building a corpus of a reasonable size.

The examples of extracted parallel sentences can be found in Table 1.

[10] `https://sourceforge.net/projects/aligner/`

Table 1: Examples of aligned sentences.

In April 1944 the Squadron moved back to the UK and re-assembled at North Weald on 23 April.	V dubnu 1944 byla peruť přeložena do Spojeného království a 23. dubna reaktivována na základně RAF North Weald.
ABAKO and Kasavubu spearheaded ethnic nationalism there and in 1956 issued a manifesto calling for immediate independence.	ABAKO a Kasavubu zde razili cestu etnickému nacionalismu a v roce 1956 vydali prohlášení volající po okamžité nezávislosti.
The band's music has a combination of influences: reggae, Latin, rock and hip hop, which is performed in a minimalistic folk style limited to vocals, beatboxing, and acoustic guitar.	Hudba 5'nizze je kombinací vlivů reggae, latinskoamerické hudby, rocku a Hip hopu v minimalistickém folkovém stylu omezeném na vokály, beatboxování a akustickou kytaru.
The Book of Abramelin tells the story of an Egyptianmage named Abramelin, or Abra-Melin, who taught a system of magic to Abraham of Worms, a GermanJew presumed to have lived from c.1362 - c.1458.	Abramelinova kniha vypráví příběh egyptského mága jménem Abramelin, nebo Abra-Melin, který předal svou nauku o magii Abrahamovi z Wormsu, německému Židu, o kterém se předpokládá, že žil v letech 1362-1458.
Her father, Kevin, is a cardiothoracic surgeon and her mother, Carolyn, was formerly an environmental engineer before becoming a homemaker.	Její otec, Kevin je kardiochirurg a její matka, Carolyn byla inženýrkou životního prostředí, než začala být ženou v domácnosti.
Accola appeared in her first movie, Pirate Camp, in 2007.	Její filmový debut přišel v roce 2007 ve filmu Pirate Camp.
Although the comet was next expected at perihelion on 1997 April, no observations were reported.	Ačkoli byla kometa znovu očekávána v periheliu roku 1997, nebyly hlášeny žádná pozorování.

After the alignment, we processed the data into vertical format required by corpus indexing system manatee and corpus manager Sketch Engine [8]: namely we a) added metadata to each document, segmented paragraphs, tokenized texts, tagged them for part of speech, prepared configuration files, prepared mapping files from the output of hunalign and compiled them.

The tools used in this chapter are available through the Natural Language Processing Centre at the Faculty of Informatics, Masaryk University[11] and at `http://corpus.tools`. E.g. unitok [9] was used for tokenization of plain texts.

Hunaling provides a score for each pair of aligned sentences. We decided to keep only the alignments with the score higher than 0.5.[12].

[11] `https://nlp.fi.muni.cz`
[12] We did not find the range of hunalign scores, the threshold was chosen heuristically.

5 Conclusion

The English corpus contains 46,238,455 tokens and the Czech corpus contains 18,785,688 tokens. The size of the aligned content is then 7,275,092 words in the English corpus and 6,414,841 words in the Czech one.

The corpus was made public and it is available at the Sketch Engine site of the Faculty of Informatics. The individual Czech and English corpora can be found under the names "Czech Wikipedia Parallel Corpus" and "English Wikipedia Parallel Corpus". The corpora are published under the CC BY-SA 4.0 license[13].

The corpus is accessible for all students and members of staff of Masaryk University. It can be used both in the field of NLP and in the field of linguistics, providing information about the language to teachers, lexicographers and translators.

Acknowledgments. This work has been partly supported by the Masaryk University within the project *Čeština v jednotě synchronie a diachronie – 2016* (MUNI/A/0863/2015) and by the Ministry of Education of CR within the LINDAT-Clarin project LM2015071.

References

1. Davies, M.: The Wikipedia corpus: 4.6 million articles, 1.9 billion words. Adapted from Wikipedia. Accessed February **15** (2015)
2. Chu, C., Nakazawa, T., Kurohashi, S.: Constructing a Chinese-Japanese Parallel Corpus from Wikipedia. In Calzolari, N., 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) (may 2014)
3. Mohammadi, M., GhasemAghaee, N.: Building bilingual parallel corpora based on wikipedia. In: Computer Engineering and Applications (ICCEA), 2010 Second International Conference on. Volume 2., IEEE (2010) 264–268
4. Pomikálek, J.: Removing boilerplate and duplicate content from web corpora. PhD thesis, Masaryk university, Faculty of informatics, Brno, Czech Republic (2011)
5. Varga, D., Németh, L., Halácsy, P., Kornai, A., Trón, V., Nagy, V.: Parallel corpora for medium density languages. In: Proceedings of the RANLP 2005. (2005) 590–596
6. Sennrich, R., Volk, M.: Mt-based sentence alignment for ocr-generated parallel texts. In: The Ninth Conference of the Association for Machine Translation in the Americas (AMTA 2010), Denver, Colorado. (2010)
7. Braune, F., Fraser, A.: Improved unsupervised sentence alignment for symmetrical and asymmetrical parallel corpora. In: Proceedings of the 23rd International Conference on Computational Linguistics: Posters, Association for Computational Linguistics (2010) 81–89
8. 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
9. Michelfeit, J., Pomikálek, J., Suchomel, V.: Text tokenisation using unitok. In Horák, A., Rychlý, P., eds.: RASLAN 2014, Brno, Czech Republic, Tribun EU (2014) 71–75

[13] `https://creativecommons.org/licenses/by-sa/4.0/`

Comparison of High-Frequency Nouns from the Perspective of Large Corpora

Maria Khokhlova

Saint-Petersburg State University,
Universitetskaya nab. 7-9-11, 199034, Saint-Petersburg, Russia
m.khokhlova@spbu.ru

Abstract. Since the last decade a number of corpora have become available, a large part of them have been compiled automatically on web data. From traditional text collections such corpora vary both in their volume and content. The paper focuses on the discussion on these corpora and deals with two of them: ruTenTen (18.3 bln tokens) and Araneum Russicum Maximum (13.7 bln tokens). The authors discuss linguistic phenomena across the corpora examining quantitative properties of 20 high-frequency Russian nouns. The lexemes are compared between these corpora and also with data published in the Frequency Dictionary on their rank distributions. This dictionary was compiled on the subset of Russian National Corpus that represents modern Russian of the 20 th century (1950–2007) and can be viewed as an excellent example of a traditional corpus. The analysis shows promising results; there is a close correlation between traditional and web-corpora and this topic should be studied in more detail paying attention to other parts of speech.

Key words: Web corpora, big data, frequency, correlation analysis, corpus evaluation

1 Introduction

Corpus linguistics, a branch of linguistics that deals with building corpora and investigation of their data, has already celebrated its 55th anniversary counting from the appearance of the Brown corpus. The idea of corpora that contain big data has attracted scholars' attention for a long time. During the last decade more and more corpora are being compiled automatically. From traditional text collections they vary both in their volume and content. This is closely related to the growing availability of technical resources and thus the gradually changing paradigm in corpus linguistics moving forward from "manual" approach to more automatic one. By a classical or traditional approach one can understand a compilation of corpora based on a previously described methodology: selection of texts involving their representativeness and balance, their correction, annotation and upload. New corpora contain in general texts that were automatically crawled from the Web. Researchers find it attractive to make statistical inferences on increasingly larger scope of data. At the same time access to large corpora provokes new challenges: what can we see with big data and how does it affect the results? Are there differences between traditional and automatically compiled corpora? The paper is organized as follows. In the next section we give an overview of the related work. In Section 3 we describe our data, while Section 4 presents the experiments and results of the analysis. Finally, Section 5 closes the paper with conclusions and suggestions for future work.

2 Related work

Large corpora with volumes exceeding 100 mln tokens have appeared just recently. The idea of creating such large text collections Nowadays one can speak about two types of corpora, some authors distinguish between three types [1]. For the Russian language the most famous and popular corpus of the first type is the National Russian Corpus; altogether its subcorpora comprise 600 mln words. This corpus can be named a traditional one and was built according to the “classic” style, i.e. linguists selected relevant texts, annotated them and included into the database. Corpora of the second type are collected automatically from the Web (obviously, to a certain degree that holds true for the first type also). For the Russian language we can name the Aranea project, which includes a few Russian corpora that differ in their size and texts among them Araneum Russicum Maximum [2]. The TenTen family [3] includes corpora of various languages of the order of 10 billion words. The ruTenTen Russian corpus is one of the biggest among them along with the English, German, French and Spanish collections. Building these corpora implies that special attention is paid to the process of de-duplication in order to delete multiple copies of the same chunks of texts. Here we leave aside rather large collections of texts that can’t be viewed as electronic corpora from the traditional viewpoint (for example, the service Google Books). To our best knowledge, there are no large corpora studies of linguistic phenomena on the Russian data, which would come up with a comparative analysis of these corpora (e.g. “big” vs. “little” corpora or “manual” vs. “automatic”). In [4] the authors present their results on studying rare Russian idioms in large corpora.

3 Data and methods

The aim of our research is to compare linguistic phenomena across different large corpora and dictionary, to identify differences, and to analyze them. We selected above mentioned two corpora that had been collected and built automatically – ruTenTen (18.3 bln tokens) and Araneum Russicum Maximum (13.7 bln tokens). In our study we used the Frequency Dictionary [5]. This dictionary was compiled on the subset of 92 mln tokens from Russian National Corpus that represents modern Russian of the 20 th century (1950–2007). It includes texts of various genres: fiction, social and political journalism, non-fiction (textbooks, social media, advertisements, technical literature) etc. The majority of Russian texts in web corpora come from news websites, blogs, commercial websites, social media groups etc. Fiction texts are less common for such corpora; therefore, we decided to focus on high-frequency vocabulary that is associated with the above-mentioned functional styles. To this end, we compiled a word list of lemmas based on the Frequency Dictionary [5]. To succeed in our study we studied frequency properties of high-frequency nouns that had been selected from the dictionary among these corpora. As nonparametric measure of statistical dependence between our data we chose Spearman's rank correlation coefficient.

4 Experiments

We compiled two lists of nouns extracted from [5] that are typical for non-fiction (see Table 1) and social and political journalism texts (see Table 2)[1]. Each list contains 20 nouns that were ranked top by frequency the Frequency Dictionary.

Table 1: High-frequency nouns in non-fiction texts

	Lemma	Translation	Frequency (ipm)
1	god	year	4624.2
2	vremja	time	2080.5
3	čelovek	man, person	1945.3
4	sistema	system	1798.0
5	rabota	job, work	1766.4
6	stat'ja	article, clause	1363.0
7	delo	affair, business	1339.5
8	slučaj	case	1259.0
9	process	process	1221.8
10	vopros	question	1180.9
11	lico	face, person	1175.9
12	sud	court	1153.9
13	čast'	part	1153.8
14	vid	kind, aspect	1147.9
15	reshenie	decision	1122.3
16	pravo	right	1117.6
17	rebënok	baby, child	1078.4
18	otnošenie	relation	1077.5
19	razvitie	development	1059.6
20	federacija	federation	1003.1

Tables 1 and 2 show that some words are shared by both lists; they belong to the high-frequency lexemes that do not depend on the genre: *čelovek* 'man, person', *delo* 'affair, business', *god* 'year', *rabota* 'job, work', *slučaj* 'case', *vopros*

[1] The Frequency Dictionary provides separate frequency lists for both types of texts.

Table 2: High-frequency nouns in texts belonging to social and political journalism

	Lemma	Translation	Frequency (ipm)
1	god	year	5589.5
2	čelovek	man, person	2950.1
3	vremja	time	2364.6
4	žizn′	life	1548.4
5	delo	affair, business	1482.0
6	den′	day	1397.8
7	rabota	job, work	1272.4
8	strana	country	1203.9
9	vopros	question	992.0
10	slovo	word	989.7
11	mesto	place	976.1
12	mir	world, peace	887.8
13	dom	house, home	879.7
14	drug	friend	850.9
15	slučaj	case	744.3
16	gorod	city, town	738.5
17	ruka	arm, hand	713.0
18	vlast′	power	711.8
19	konec	end	710.8
20	sila	strength	709.8

'question', *vremja* 'time'. It is worth mentioning that the average frequency of the nouns presented in Table 1 is higher than in Table 2. It can be supposed that the given high-frequency lexemes are more often used in non-fiction texts than in newspapers (however the volume of non-fiction subcorpus is less). In our research we have also analyzed 20 top-frequency nouns in the two corpora. The following list was compiled for ruTenTen corpus: *god* 'year', *rabota* 'job, work', *vremja* 'time', *čelovek* 'man, person', *kompanija* 'company', *sistema* 'system', *sajt* 'site', *den′* 'day', *mesto* 'place', *Rossija* 'Russia', *vid* 'kind, aspect', *vopros* 'question', *slučaj* 'case', *rebënok* 'baby, child', *žizn′* 'life', *vozmožnost′* 'opportunity, possibility', *kačestvo* 'quality', *programma* 'programme', *delo* 'affair, business', *usluga* 'service, favour'. For Araneum Russicum Maximum corpus the following nouns were most frequent: *god* 'year', *čelovek* 'man, person', *vremja* 'time', *rabota* 'job', *den′* 'day', *kompanija* 'company', *oblast′* 'region, field', *sistema* 'system', *sajt* 'site', *mesto* 'place', *vopros* 'question', *žizn′* 'life', *slučaj* 'case', *Rossija* 'Russia', *vid* 'kind, aspect', *dom* 'house, home', *delo* 'affair, business', *strana* 'country', *raz* 'time, one', *vozmožnost′* 'opportunity, possibility'. We can see that a half of the first list overlaps with Table 2 whereas twelve nouns from the second list coincide with the data in the same table. Based on this preliminary analysis of the lists we can see that two corpora share more in common with newspapers than with non-fiction texts. Comparing two lists it can be said that the majority of the nouns presents in both of them that indicates the similarity of the corpora. Lexemes *sajt* 'site' and *kompanija* 'company' were not among 20 most frequent nouns selected from the Frequency Dictionary but were ranked top by frequency in the lists for both corpora. This fact can be explained on one hand that a vast number of data for the corpora is crawled from news web-sites and on the other hand a lot of texts have description of web pages and their content. This holds particularly true for ruTenTen corpus due to other frequent lexemes: *vozmožnost'* 'opportunity, possibility', *kačestvo* 'quality', *programma* 'programme', *usluga* 'service, favour'. We referred to the two corpora to study frequencies of the words on the lists (see Tables 1 and 2); you can find the results on Table 3 and Fig. 1. as well as on Table 4 and Fig. 2.

Table 3: Frequencies of nouns on the non-fiction word list (journalism excluded) calculated as per two corpora

	Lemma	Translation	Frequency word list for non-fiction in the Frequency Dictionary	ruTenTen	Araneum Russicum Maximum
1	god	year	4624.2	3080.0	3263.0
2	vremja	time	2080.5	1791.0	1857.0
3	čelovek	man, person	1945.3	1956.0	2012.0
4	sistema	system	1798.0	999.0	1011.0
5	rabota	job, work	1766.4	1510.0	1632.0
6	stat'ja	article, clause	1363.0	294.1	446.8
7	delo	affair, business	1339.5	814.0	741.0
8	slučaj	case	1259.0	752.0	758.0
9	process	process	1221.8	474.0	491.9
10	vopros	question	1180.9	866.0	855.0
11	lico	face, person	1175.9	483.7	458.1
12	sud	court	1153.9	303.2	255.7
13	čast'	part	1153.8	677.0	650.3
14	vid	kind, aspect	1147.9	723.0	806.0
15	reshenie	decision	1122.3	558.0	556.3
16	pravo	right	1117.6	507.2	405.1
17	rebënok	baby, child	1078.4	850.0	443.1
18	otnošenie	relation	1077.5	481.2	438.4
19	razvitie	development	1059.6	587.0	570.6
20	federacija	federation	1003.1	198.4	168.0

Table 3 and Fig. 1 show the data for nouns in Table 1. We can see that both corpora show similar curves on the graph, which means that these words have similar distribution. Both corpora agree on the ranking of the seven words five of them being on the top of the lists. Spearman's rank correlation coefficient between the ranked word lists of ruTenTen and Araneum Russicum Maximum corpora is 0.89 that indicates a very high correlation. The rank coefficient between the lists in the Frequency Dictionary and in ruTenTen is 0.63, whereas the coefficient between the Frequency Dictionary and the Araneum Russicum Maximum corpus stands at 0.76, which in the latter case reveals that the dictionary and the corpus have much more in common. The frequencies, indicated in the Dictionary, are the highest, except the frequency of the lemma *čelovek* 'man, person' which has the highest frequency in ruTenTen. Two corpora rank the words differently from the ranking in the Dictionary – two nouns in ruTenTen have the same ranking as in the Dictionary, and Araneum Russicum Maximum contains three such nouns.

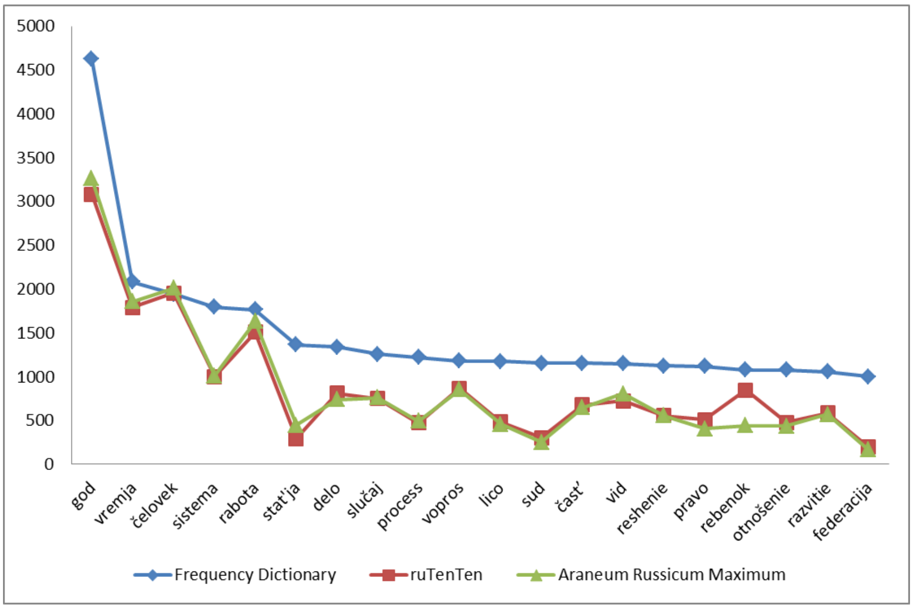


Fig. 1: Frequency distribution of nouns on the non-fiction word list (journalism excluded) as per two corpora (x-axis: nouns; y-axis: frequency in ipm)

On Fig. 2 we can see the data for the nouns in Table 2; like the results on Fig. 1. Fig 2 shows that both the ruTenTen and Araneum Russicum Maximum corpora yield to a certain degree identical results. The word *rabota* 'job, work' (as well as *slučaj* 'case' and *gorod* 'city, town', see Table 4) has higher frequency in Araneum Russicum Maximum, than in the Dictionary; for other nouns the Dictionary shows maximum frequency values. Four nouns have identical rankings in the Frequency Dictionary and both in ruTenTen and Araneum Russicum Maximum corpora. In case of two corpora the number of such nouns (that have the same ranks) is nine. Spearman's rank correlation coefficient between the ranked word lists in the Frequency Dictionary and in ruTenTen is high standing at 0.84 and it is 0.82 for the word lists in the Frequency Dictionary and in Araneum Russicum Maximum. This can indicate that both corpora more in common with newspaper articles and similar texts and moreover with Russian National Corpus (as it was the source for the Frequency Dictionary). Spearman's rank correlation coefficient between the lists in two corpora is remarkably large and equals 1 (this points to the highest correlation).

Table 4: Frequencies of nouns on the social and political journalism word list as per the two corpora

	Lemma	Translation	Social & political journalism word list in the Frequency Dictionary	ruTenTen	Araneum Russicum Maximum
1	god	year	5589.50	3080.0	3263.0
2	čelovek	man, person	2950.10	1956.0	2012.0
3	vremja	time	2364.60	1791.0	1857.0
4	žizn′	life	1548.40	865.0	899.0
5	delo	affair, business	1482.00	814.0	741.0
6	den′	day	1397.80	1089.0	1253.0
7	rabota	job, work	1272.40	1510.0	1632.0
8	strana	country	1203.90	662.0	657.6
9	vopros	question	992.00	866.0	855.0
10	slovo	word	989.70	645.0	563.3
11	mesto	place	976.10	950.0	970.0
12	mir	world, peace	887.80	626.0	655.5
13	dom	house, home	879.70	689.0	751.0
14	drug	friend	850.90	452.3	500.7
15	slučaj	case	744.30	752.0	758.0
16	gorod	city, town	738.50	757.0	792.0
17	ruka	arm, hand	713.00	466.7	430.5
18	vlast′	power	711.80	330.0	273.9
19	konec	end	710.80	417.8	344.4
20	sila	strength	709.80	467.5	438.2

5 Conclusion and Further Work

We come to the general conclusion that texts selected for large corpora feature the language of the web and their structure corresponds to newspaper texts and thus to journalistic genre. The Araneum Russicum Maximum appears to be slightly more consistent with the Frequency Dictionary than the ruTenTen corpus in describing high-frequency nouns. For the given high-frequency nouns there is a very strong association between the data obtained on two corpora. Hence it can be supposed that there is no difference between the

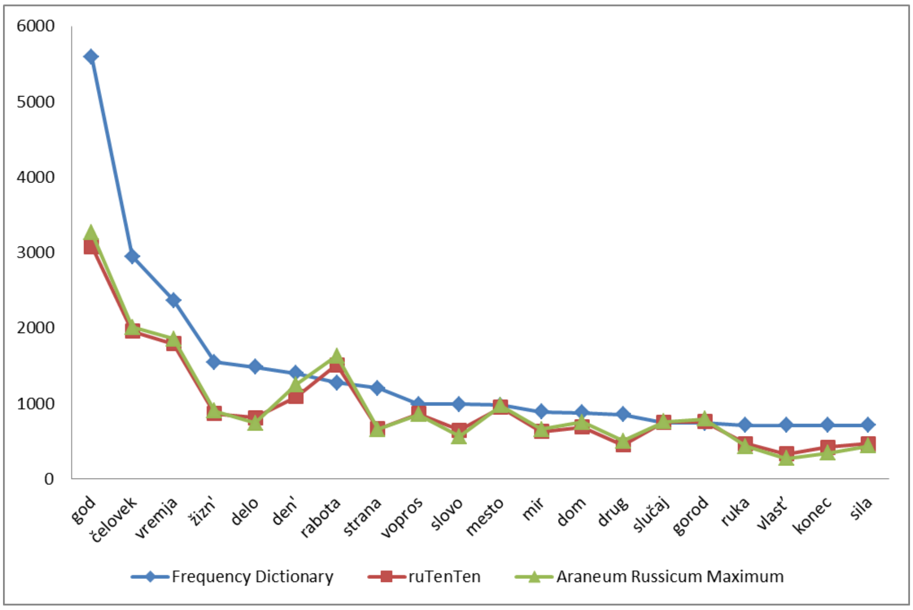


Fig. 2: Frequency distribution of nouns on the social and political journalism word list as per two corpora (x-axis: nouns; y-axis: frequency in ipm)

automatically crawled corpora in case of high-frequency lexemes. Both corpora show quite a high correspondence with the Frequency dictionary. The data selected from the Frequency dictionary were based on the Russian National Corpus and therefore the obtained results reveal a close correlation between traditional and web-corpora. Our next work will be targeted at other parts of speech as nouns can be thematically biased, and their frequencies can depend on types of texts and thus differ dramatically even among corpora compiled within the same methodology.

Acknowledgments. This work was supported by the grant of the President of Russian Federation for state support of scholarly research by young scholars (Project No. MK-5274.2016.6).

References

1. Belikov, V., Selegey, V., Sharoff, S.: Prolegomeny k proyektu General'nogo internet-korpusa russkogo yazyka (GIKRYa) [Preliminary considerations towards developing the General Internet Corpus of Russian]. In Computational linguistics and intellectual technologies. Vol. 11 (18), pp. 37-49. RSUH, Moscow (2012).
2. Benko, V.: Aranea: Yet Another Family of (Comparable) Web Corpora. In Text. Speech and Dialogue. 17th International Conference. TSD 2014, pp. 257-264. Springer, Switzerland (2014).

3. Jakubíček, M., Kilgarriff, A., Kovář, V., Rychlý, P., Suchomel, V.: The TenTen Corpus Family. In Proceedings of the International Conference on Corpus Linguistics, pp. 125-127. Lancaster, UK (2013).
4. Benko, V., Zakharov, V.: Very Large Russian Corpora: New Opportunities and New Challenges. In Computational linguistics and intellectual technologies. Vol. 15 (22), pp. 79-93, RSUH, Moscow (2016).
5. Lyashevskaya, O., Sharoff, S.: Častotnyj slovar′ sovremennogo russkogo jazyka (na materialax Nacional′nogo Korpusa Russkogo Jazyka) [Frequency Dictionary of Contemporary Russian based on the Russian National Corpus data]. Azbukovnik, Moscow (2009).

Feeding the "Brno Pipeline": The Case of Araneum Slovacum

Vladimír Benko[1,2]

[1] Slovak Academy of Sciences, Ľ. Štúr Institute of Linguistics
Panská 26, SK-81101 Bratislava, Slovakia
[2] Comenius University in Bratislava, UNESCO Chair in Plurilingual and Multicultural Communication
Šafárikovo nám. 6, SK-81499 Bratislava, Slovakia
`vladob@juls.savba.sk`
`http://www.juls.savba.sk/~vladob`

Abstract. Our paper is targeted at our experiences with a set of tools developed at the Masaryk University in Brno Faculty of Informatics, and it presents a case study describing our Autumn 2016 web crawl and its subsequent processing for the *Araneum Slovacum Maximum* web corpus.

Key words: web-based corpora, web data filtration and deduplication, Aranea Project

1 Introduction

The "Brno Pipeline" (BPL) is a term coined by Nikola Ljubešić [7] for a complete set of tools developed at Masaryk University in Brno, Faculty of Informatics that provide for the effective creation of corpora based on web-derived data. If complemented by an appropriate morphosyntactic tagger, BPL basically covers the whole process, practically without any need for additional programming capacity. Our work is a case study describing the use of BPL for the creation of *Araneum Slovacum Maximum* Slovak web corpus that is continuously being built since 2013 within the framework of the Aranea Project aimed at the creation of a family of comparable web corpora [3,4]. We will show some general considerations, describe the parameters of the whole process, and introduce some tools of our own implementing additional functionality not provided by BPL itself.

2 Data Crawling

The main BPL component is SpiderLing[3], a crawler highly optimized for downloading textual data from the web [12]. The tool also integrates modules

[3] `http://corpus.tools/wiki/SpiderLing`

for web page encoding detection (Chared[4]), boilerplate removal (jusText[5]; [8]), trigram-based language identification (trigrams[6]), and detection and removal of identical documents. The user-supplied input consists of a text sample for the respective language to be used to build a model for the language identification procedure, and a set of seed URL addresses needed for bootstrapping the crawl process.

Our experience manifests that the quality of the sample text is worth manual checking, as text fragments with non-standard orthography or those in foreign language(s) negatively influence the results of the language detection, resulting in large quantities of unwanted texts at the output of the procedure.

Within our Aranea Project, several methods of collecting URLs had been tested. The most convenient proved to be using the BootCaT[7] program [1] in the "seed words" mode. This may be quite simple if a PoS-tagged frequency word list is available. Such a list could be easily obtained from the Slovak National Corpus or, as in our case, from the already existing version of Araneum Slovacum. We decided to use the list of 1,000 most frequent adverbs, which is the word class with rather general meaning and almost no inflection. The list was used for extraction of randomly chosen groups of 20 words that were subsequently submitted as seeds for BootCaT. During each BootCaT run, 200 triples were created to be submitted as search expressions for the Bing[8] search engine requiring retrieval of the maximum count of 50 URLs during each search. Our typical BootCaT session consisted of 5 runs, theoretically yielding as many as 50,000 URLs. This count, however, usually dropped down to some 25,000 to 45,000 after removing duplicate URLs, as well as those pointing to wrong document types (such as PDFs that could not be processed by SpiderLing).

Probably the most important issue in using SpiderLing is its consumption of RAM – the author(s) seem to have expected that only very powerful server configurations would be utilized. As all necessary data structures storing information on the visited web pages, as well as on "wrong" URLs and duplicates, are kept in the main memory, the crawling process "freezes" when allocation of more RAM is not possible. In our case we could afford to dedicate for the long-time crawling only a machine with 16 GB of RAM, which usually led to freezing after some 80 hours of SpiderLing operation. The new crawling had to be started with a fresh set of seed URLs from scratch, risking the potentially high amount of duplicates appearing in the crawled data.

The RAM consumption, however, can be influenced by several user-settable parameters, with one of them being the restriction on top-level domain (TLD)

[4] `http://corpus.tools/wiki/Chared`
[5] `http://corpus.tools/wiki/Justext`
[6] `http://code.activestate.com/recipes/326576-language-detection-using-character-trigrams/`
[7] `http://bootcat.sslmit.unibo.it/`
[8] `http://www.bing.com/`

of the crawled documents. No restriction on TLDs results in an increase of RAM consumption, which may not be quite an intuitive behaviour.

During the SpiderLing operation, exact duplicates are identified (but not removed) on fly. Removal of the dupes can be performed at the end of a crawling session by a simple script. The resulting text file is in a "one line per paragraph" format containing light-weight XML markup describing docs, paragraphs and (optionally) the deleted boilerplate data.

3 Pre-Tokenization Filtration

Filtration aims at removing the documents containing texts that do not adhere to a predefined quality standard, and also those containing (partially) duplicate contents. This process is (at least in part) language-dependent, so we have written a series of filters that are being sequentially applied to the source data. Our filters typically consist of two components – the analyser generates a list of documents with a certain parameter above/under the specified threshold, and the removal procedure uses this list to split the input file into two parts, one containing the "good" docs and the other the "bad" ones. The advantage of such implementation is that the removal procedure can be universal, i.e., filter-independent, and also the fact that the removed documents can be subsequently analysed to provide for optimizing the parameters of filtration. Most of our own tools have been written in a rather "vintage" programming language, flex based on regular expressions and C code. The disadvantage of this approach is that flex is not compliant with Unicode (UTF-8), i.e., all computations have to be performed on the byte level only, and multi-byte UTF-8 characters must be treated by the programmer him/herself. On the other hand, flex programs tend to execute (at least) by order of magnitude faster than those written in an interpreted language such as Python and the actual speed of a filter is typically on par with plain file copying.

As we usually work with very large files, the sequence of filters can be conveniently optimized in order to remove most of the "wrong" data during the first step(s). The optimal sequence, however, is usually language-dependent, and in case of Slovak it is as follows:

1. Identification and removal of "insufficiently Slovak" documents. As the trigram language identification module is usually not able to cope with the differences among languages with similar character frequency distributions, not only lots of Czech texts appear in the data, but also some Croatian, Serbian, Slovene texts can be seen there. Our supplementary filter is based on counting the average frequencies of Slovak letters with diacritics, and separately counting two special cases: the missing "š/Š" a "ž/Ž" usually indicate an encoding issue (mostly Windows 1250 encoding misidentified as ISO 8859-2), while missing "ľ/Ľ" may mean that it is in fact a Czech text.

2. Identification and removal of exact duplicates by the fingerprint method [2]. These could not have been removed by SpiderLing itself, if sev-

eral independent crawling sessions had been performed. (The partial duplicates would be removed after tokenization only.)

3. Identification and removal of “too Czech” documents. Despite having passed the Slovak filter, some documents may nonetheless contain Czech text fragments. An algorithm analogous to that of Slovak is used, with the main difference being that the characters “ě/Ě”, “ř/Ř” and “ů/Ů” (not present in Slovak) are counted.

4. Identification and removal of documents with incorrectly interpreted encoding, containing artefacts such as “po hrebeĹˆoch hĂ′r ÄŒeska, Slovenska a PoÄžska” (instead of “po hrebeňoch hôr Česka, Slovenska a Poľska”) caused by treating a UTF-8 as 8-bit encoding. Quite often only a small fragment of a document may be affected. We, however, prefer removing such documents as a whole.

4 Tokenization

A standard BPL tool for tokenization is unitok [9], complemented by a language-specific parameter file. As no Slovak parameter file was present in the standard unitok distribution, we have created a new one based on the analogous Czech file. The new contents consist mainly of a list of period-final abbreviations, partially translated from the Czech list, and subsequently updated by abbreviations actually found in the Slovak corpus data. The only major tokenization policy change was the decision of tokenizing the “multi-period” abbreviations such as “s.r.o.” as three separate tokens, so that it would be more compatible with the language model used by the tagger.

As the Python code takes much longer to execute than the flex filters, we usually run the tokenization as 4 processes in parallel to make use of the multicore processor of our server.

5 Post-Tokenization Filtration

Some encoding and other issues are easier to detect in an already tokenized text, as the regular expressions can rely on correct word boundaries. This is why some filters are better run after tokenization only. One of such filters is detecting situations defined as “uppercase letter with diacritics inside otherwise lowercase word”. There may be several causes of this a phenomenon, such as a typo – incorrectly pressed SHIFT key (“vŠetko”, “antikvariÁt”), “lost” spaces between two words (“vŽiline”, “voŠvajciarsku”), incorrect interpretation of encoding (“veŸmi”, “zÄava”), or even corrupted HTML entities (“nbspŠali”). We must, however, be cautious here – some of the tokens detected by this simplistic approach could represent legitimate neologisms with non-standard orthography (“eŠkola”, “eČajovňa”).

Table 1: Explanation of *ztag*.

0	The word has not been found in the lexicon, lemma has been just copied from word form.
1	The word form has been found in the lexicon and the lemma has been assigned unambiguously.
2, 3, 4	The word form has been found in the lexicon with 2 to 4-way ambiguity. Lemma contains all possible variants separated by a vertical line (“ | ”).

6 Detection of Near-Duplicate Documents

The next important processing step is deduplication. We can conveniently use another BPL component here – the Onion tool[9] [8]. The principle of its operation and testing its various settings was treated in our work [2]. We only mention two parameters here: deduplication is performed on 5-grams with similarity threshold level 0.9.

7 Morphosyntactic Annotation

The tagging process is mentioned only briefly here, as the tagger itself is neither part of BPL, nor our own tool, and also because annotation deserves a paper of its own. Besides the use of the tagger itself, our morphosyntactic annotation involves several additional steps: pre-tagging and post-tagging filters performing the “lemmatization” of punctuation and special graphic characters, marking the out-of-vocabulary (OOV) tokens and mapping the “native” tags universal PoS-only tags.

Our Slovak corpora are currently being tagged by TreeTagger[10] [11], using our own language model trained on the manually disambiguated *rmak-4.0* Slovak corpus [11] and the updated SNC morphological database using the SNC tagset [5].

TreeTagger also implements a guesser assigning tags to tokens not found in the lexicon. However, it does not try to guess lemmas for such tokens. In our Aranea corpora, the special attribute *ztag* is used to indicate the result of tagging, see Table 1 for explanation.

The tag attribute can be conveniently used in analysing the results of tagging, as problematic phenomena in the corpus can be queried explicitely.

The pre-tagging and post-tagging filter modify the lemmas and tags for punctuation and special graphic characters, providing what we may call a

[9] `http://corpus.tools/wiki/Onion`

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

[11] `http://korpus.sk/ver_r(2d)mak.html`

Table 2: Resulting vertical file.

word	lemma	atag	tag	ztag
Dropbox	Dropbox	Nn	SSis4	0
i	i	Cj	O	1
SkyDrive	SkyDrive	Nn	SSns6	0
buď	budiť | byť	Vb	VMesb+	2
inštalujete	inštalovať	Vb	VKjpb+	1
,	,	Zz	Z	1
alebo	alebo	Cj	O	1
používate	používať	Vb	VKepb+	1
jeho	jeho	Pn	PUfs2	1
webového	webový	Aj	AAms2x	1
klienta	klient	Nn	SSms2	1
.	.	Zz	Z.	1

"lemmatization". For example, several Unicode representations of an apostrophe are retained as "word forms", but mapped to an "ASCII apostrophe" at the lemma level.

Similarly to other Aranea corpora, the "native" tags are mapped to Araneum Universal Tagset (AUT)[12], providing a parallel level of annotation. The respective values from the AUT tagset in the Aranea corpora are stored in the atag attribute.

The resulting vertical file after all annotation steps contains five attributes: *word, lemma, atag, tag* and *ztag*, see for example Table 2.

We can see two cases of an OOV item in our sentence, as well as a case of a 2-way ambiguous lemma – both variants, however, being incorrect in this particular case.

8 Paragraph-level Deduplication

Our corpus data can be utilized both as source data for various NLP projects or in a traditional way for "manual" analysis by means of a corpus manager. Depending on the mode of use, we implement two different policies of paragraph-level deduplication. For the NLP use, where nobody is expected to analyse the data by "reading" it, we prefer that the dupe paragraphs be deleted. For traditional work with a corpus manager we do not want to "destroy" the cohesion of the text by "randomly" deleting paragraphs inside a document. In this case we only mark the dupes so that they do not appear in the results of

[12] `http://aranea.juls.savba.sk/aranea_about/aut.html`

query operations, yet they could be displayed at the boundary of duplicate and non-duplicate text.

In both cases we use Onion with standard settings: 5-grams with similarity threshold 0.9.

9 Corpus Managers

The Araneum Slovacum is used in our Institute by lexicographers within our local installation of Sketch Engine[13] [6]. It is, however, available also for the general public at the NoSketch Engine[14] [10] Aranea Project portal.

10 The Autumn 2016 *Araneum Slovacum Maximum* Crawl

This section brings some data on our latest crawl and processing session performed in October 2016. The respective processing steps are shown in Table 3 and are accompanied by the relevant data on sizes and times. The crawling process itself consisting of six separate sessions is not included in the table. During this crawl we decided to experiment with releasing the TLD restriction.

As seen in Table 3, our decision not to limit the TLDs of the crawled web pages caused a large amount of "insufficiently Slovak" texts being removed by the very first filter. A brief checking reveals that most of the removed texts are Czech. They are not going to be disposed – we can use them during the next upgrade of our Czech *Araneum Bohemicum Corpus*. The bottom line, however, is that not setting TSD was probably not a good decision.

11 Conclusion and Further Work

After using the BPL tools for a fairly long time we can say that they represent a mature, efficient and robust set providing for all main procedures necessary to build a web corpus of our own. This also means that, having spare programming resources at hand, these can be targeted to language-dependent filtration and tiny improvements of the whole process. We can also see that implementation of the supplementary tools in flex tends to cut significantly the processing times, which is also the main reason why we have not decided yet to rewrite them into Python.

Our next plans – besides the fine-tuning of the whole process – includes testing some alternative tools, most notably the taggers.

Acknowledgments. The presented results were partially obtained under the VEGA Grant Agency Project No. 2/0015/14 (2014–2016).

[13] https://www.sketchengine.co.uk/

[14] https://nlp.fi.muni.cz/trac/noske

Table 3: Crawling results.

Operation	Output	Processing time (hh:mm)
Merging crawled text data from six SpiderLing sessions, assigning document Ids, fixing minor URL issues introduced by SpiderLing markup	7,108,601 docs 35.99 GB	n/a
Identifying and removing "insufficiently Slovak" documents	1,775,619 docs (75.02% removed) 9.41 GB	0:20
Identifying and removing exact duplicates by fingerprint method	1,370,075 docs (22.84% removed) 7.37 GB	0:17
Removing survived HTML markup and normalizing encoding (Unicode spaces, composite accents, soft hyphens, etc.)	7.36 GB	0:06
Removing successive duplicate paragraphs (by uniq)	7.31 GB	0:05
Identifying and removing "too Czech" documents	1,276,592 docs (6.82% removed) 6.51 GB	0:04
Identifying and removing documents with encoding issues	1,272,622 docs (0.31% removed) 6.49 GB	0:03
Tokenization by Unitok (4 parallel processes, custom Slovak parameter file)	980,058,957 tokens 7.39 GB	1:56
Truncating long tokens	7.39 GB	0:05
Identifying and removing documents with encoding issues (bis)	1,269,852 docs (0.22% removed) 7.36 GB	0:04
Segmenting to sentences (rudimentary rule-based algorithm)	56,969,058 sents 7.87 GB	0:06
Identifying and removing partially identical documents by Onion (5-grams, similarity threshold 0.9)	754,360 docs 559,387,978 tokens (42.63% removed) 4.50 GB	0:27
Pre-tagging filtration of punctuation and special graphic characters		0:03
Tagging by Tree Tagger with custom Slovak language model (4 parallel processes)	10.60 GB	0:56
Restoring original wordforms, marking the out-of-vocabulary (OOV) tokens (ztag), mapping native SNK tags to "PoS-only" AUT tagset (atag).	82,786,567 tokens marked OOV (6.43%)	0:08
Identifying and removing or marking partially duplicate paragraphs by Onion	(not performed yet at present)	0:0

References

1. Baroni, B., Bernardini, S.: BootCaT: Bootstrapping corpora and terms from the web. In: Proc. 4th Int. Conf. on Language Resources and Evaluation, Lisbon : ELRA (2004)
2. Benko, V: Data Deduplication in Slovak Corpora. In: Slovko 2013: Natural Language Processing, Corpus Linguistics, E-learning, pp. 27–39. RAM-Verlag, Lüdenscheid (2013)
3. Benko, V.: Aranea: Yet another Family of (Comparable) Web Corpora. In: Text, Speech, and Dialogue. 17th International Conference, TSD 2014 Brno, Czech Republic, September 8—12. Proceedings. Eds. P. Sojka et al. Cham – Heidelberg – New York – Dordrecht – London : Springer, pp. 21—29. ISBN 978-3-319-10816-2. (2014)
4. Benko, V.: Two Years of Aranea: Increasing Counts and Tuning the Pipeline. In Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC). Portorož : European Language Resources Association (2016) pp. 4245–4248. ISBN 978-2-9517408-9-1. (2016)
5. Garabík, R., Šimková, M.: Slovak Morphosyntactic Tagset. Journal of Language Modelling, 0(1), pp. 41–63 (2012)
6. Kilgarriff, A., Rychlý, P., Smrž, P., Tugwell, D.: The Sketch Engine. In: Proc. XI EURALEX Int. Congress, Lorient, pp. 105–116 (2004)
7. Ljubešić, N., Klubička, F.: {bs,hr,sr}WaC – Web corpora of Bosnian, Croatian and Serbian. Proceedings of the 9th Web as Corpus Workshop (WaC-9). Gothenburg, Sweden. (2014)
8. Pomikálek, J.: Removing Boilerplate and Duplicate Content from Web Corpora. Ph.D. thesis, Masaryk University, Brno (2011)
9. Michelfeit, J., Pomkálek, J., Suchomel, V.: Text Tokenisation Using unitok. In Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Ad-vances in Slavonic Natural Language Processing, RASLAN 2014, pp. 71—75, 2014. Brno: NLP Consulting (2014)
10. Rychlý, P.: Manatee/Bonito – A Modular Corpus Manager. In: 1st Workshop on Recent Advances in Slavonic Natural Language Processing. pp. 65–70. Masaryk University, Brno (2007)
11. Schmid, H.: Probabilistic Part-of-Speech Tagging Using Decision Trees. In: Proceedings of International Conference on New Methods in Language Processing. Manchester (1994)
12. Suchomel V., Pomikálek J.: Efficient Web Crawling for Large Text Corpora. In: 7th Web as Corpus Workshop (WAC-7), Lyon, France, pp. 24–31. (2012)

Gold-Standard Datasets for Annotation of Slovene Computer-Mediated Communication

Tomaž Erjavec[1], Jaka Čibej[2], Špela Arhar Holdt[2,3], Nikola Ljubešić[1,4], and Darja Fišer[2,1]

[1] Department of Knowledge Technologies, Jožef Stefan Institute,
Jamova cesta 3, SI-1000 Ljubljana, Slovenia
tomaz.erjavec@ijs.si, nikola.ljubesic@ijs.si
[2] Dept. of Translation, Faculty of Arts, University of Ljubljana,
Aškerčeva cesta 2, SI-1000 Ljubljana, Slovenia
jaka.cibej@ff.uni-lj.si, spela.arharholdt@ff.uni-lj.si,
darja.fiser@ff.uni-lj.si
[3] Trojina, Institute for Applied Slovene Studies,
Trg republike 3, SI-1000 Ljubljana, Slovenia
[4] Department of Information and Communication Sciences, University of Zagreb
Ivana Lučića 3, HR-10000 Zagreb, Croatia

Abstract. This paper presents the first publicly available, manually annotated gold-standard datasets for the annotation of Slovene Computer-Mediated Communication. In this type of language, diacritics, punctuation and spaces are often omitted, and phonetic spelling and slang words frequently used, which considerably deteriorates the performance of text processing tools that were trained on standard Slovene. Janes-Norm, which contains 7,816 texts or 184,766 tokens, is a gold-standard dataset for tokenisation, sentence segmentation and word normalisation, whereas Janes-Tag, comprising 2,958 texts or 75,276 tokens, was created for training and evaluating morphosyntactic tagging and lemmatisation tools for non-standard Slovene.

Key words: Slovene language, Computer-Mediated Communication, Word Normalisation, Morphosyntactic Tagging, Lemmatisation

1 Introduction

The development of language technologies for individual languages needs hand annotated datasets for evaluation and, with machine learning methods currantly being the dominant paradigm, also for training language models for all the relevant levels of text annotation. At least for basic text annotation, tools and datasets have already been developed for Slovene: morphosyntactic tagging and lemmatisation can be performed with ToTaLe [5] and Obeliks [11], while new tools can be trained on the openly available manually annotated corpus ssj500k [13] and the morphological lexicon Sloleks [3].

However, these tools and resources predominantly deal with standard Slovene. In recent years the growing importance and quantity of Computer-Mediated Communication (CMC), such as contained in tweets and blogs, has led to a sharp increase in interest in processing such language. Tools for annotating standard language perform poorly on CMC [10], as diacritics and punctuation are often omitted, and phonetic spelling and slang words frequently used, leading to many unknown words for standard models.

The Janes[5] project aims to change this situation by developing a corpus of Slovene CMC, performing linguistic analysis on it and developing robust tools and hand-annotated gold-standard datasets for tool training and testing. It is the last goal that is the topic of this paper, which is structured as follows: Section 2 introduces the Janes corpus of Slovene CMC with an emphasis on the tools that were used to annotate it with linguistic information; Section 3 details the annotation campaign in which samples from the Janes corpus were manually annotated; Section 4 overviews the encoding, distribution and quantitative data on the resulting two datasets; and Section 5 gives some conclusions and directions for further research.

2 The Janes corpus and its annotation

The Janes corpus of Slovene CMC has been prepared in several iterations, with the current version being Janes 0.4 [8]. It contains five types of public CMC text types: tweets, forums, user comments on internet news articles (and, for completeness, also the news articles themselves), talk pages from Wikipedia and blog articles with user comments on these blogs. The collection of tweets and Wikipedia talk pages is comprehensive in the sense that the corpus includes all the users and their posts that we identified at the time of the collection. For the other texts types we selected, due to time and financial constraints, only a small set of sources that are the most popular in Slovenia and offer the most texts. Version 0.4 contains just over 9 million texts with about 200 million tokens, of which 107 come from tweets, 47 from forum posts, 34 from blogs and their comments, 15 from news comments and 5 from Wikipedia.

The texts in the corpus are structured according to the text types they belong to, e.g. conversation threads in forums, and contain rich metadata, which have been added manually (e.g. whether the author of a tweet or blog is male or female, whether the account is corporate or private) or automatically (e.g. text sentiment). For this paper, the most relevant piece of text metadata is the assignment of standardness scores to each text. We developed a method [15] to automatically classify a texts into three levels of technical and linguistic standardness. Technical standardness (T1, quite standard – T3, very non-standard) relates to the use of spaces, punctuation, capitalisation and similar, while linguistic standardness (L1 – L3) takes into account the level of adherence

[5] "Janes" stands for "Jezikoslovna analiza nestandardne slovenščine" (Linguistic Analysis of Non-Standard Slovene). The home page of the project is `http://nl.ijs.si/janes/` and the project lasts 2014–2017.

to the written norm and more or less conscious decisions to use non-standard language, involving spelling, lexis, morphology, and word order. On the basis of a manually labelled test set the method has a mean error rate of 0.45 for technical and 0.54 for linguistic standardness prediction.

The texts in the corpus have been linguistically annotated with automatic methods for five basic levels, which we describe in the remainder of this section. The tools have been developed mostly in the scope of the Janes project and typically rely on supervised machine learning. For each tool we briefly report on the training data used and, where available, the estimated accuracy of the tool.

2.1 Tokenisation and sentence segmentation

For tokenisation and sentence segmentation we used a new (Python) tool that currently covers Slovene, Croatian and Serbian [16]. Like most tokenisers, ours is based on manually specified rules (implemented as regular expressions) and uses language-specific lexicons with, e.g. lists of abbreviations. However, the tokeniser also supports the option to specify that the text to be processed is non-standard. In this case it uses rules that are less strict than those for standard language as well as several additional rules. An example of the former is that a full stop can end a sentence even though the following word does not begin with a capital letter or is even not separated from the full stop by a space. Nevertheless, tokens that end with a full stop and are on the list of abbreviations that do not end a sentence, e.g. `prof.` will not end a sentence. For the latter case, one of the additional regular expressions is devoted to recognising emoticons, e.g. `:-]`, `:-PPPP`, `^_^` etc.

A preliminary evaluation of the tool on tweets showed that sentence segmentation could still be significantly improved (86.3% accuracy), while tokenisation is relatively good (99.2%) taking into account that both tasks are very difficult for non-standard language.

2.2 Normalisation

Normalising non-standard word tokens to their standard form has two advantages. First, it becomes possible to search for a word without having to consider or be aware of all its variant spellings and, second, tools for standard language, such as part-of-speech taggers, can be used in further linguistic processing if they take as their input the normalised forms of words. In the Janes corpus all the word tokens have been manually examined and normalised when necessary by using a sequence of two steps.

Many CMC texts are written without using diacritics (e.g. `krizisce` $\rightarrow$ `križišče`), so we first use a dedicated tool [17] to restore them. The tool learns the rediacritisation model on a large collection of texts with diacritics paired with the same texts with diacritics removed. The evaluation showed that the tool achieves a token accuracy of 99.62% on standard texts (Wikipedia) and 99,12% on partially non-standard texts (tweets).

In the second step the rediacriticised word tokens are normalised with a method that is based on character-level statistical machine translation [14]. The goal of the normalisation is to translate words written in a non-standard form (e.g. `jest`, `jst`, `jas`, `js`) to their standard equivalent (`jaz`). The current translation model for Slovene was trained on a preliminary version of the manually normalised dataset of tweets presented in this paper, while the target (i.e. standard) language model was trained on the Kres balanced corpus of Slovene [18] and the tweets from the Janes corpus that were labelled as linguistically standard using the tool described above.

It should be noted that normalisation will, at times, also involve word-boundaries, i.e. cases where one non-standard word corresponds to two or more standard words or vice versa (e.g. `ne malo` → `nemalo`; `tamau` → `ta mali`). As will be shown, this raises a number of challenges both in the manual annotation and in the encoding of the final resource, as the mapping between the original tokens and their normalised versions (and their annotation) is no longer 1-1.

2.3 Tagging and lemmatisation

As the last step in the text annotation pipeline the normalised tokens are annotated with their morphosyntactic description (MSD) and lemma. For this we used a newly developed CRF-based tagger-lemmatiser that was trained for Slovene, Croatian and Serbian [16]. The main innovation of the tool is that it does not use its lexicon directly, as a constraint on possible MSDs of a word, but rather indirectly, as a source of features; it thus makes no distinction between known and unknown words. For Slovene the tool was trained on the already mentioned ssj500k 1.3 corpus [13] and the Sloleks 1.2 lexicon [3]. Compared to the previous best result for Slovene using the Obeliks tagger [11], the CRF tagger reduces the relative error by almost 25% achieving 94.3% on the testing set comprising the last tenth of the ssj500k corpus.

It should be noted that the MSD tagset used in Janes follows the (draft) MULTEXT-East Version 5 morphosyntactic specifications for Slovene[6], which are identical with the Version 4 specifications [6], except that they, following [1], introduce new MSDs for annotation of CMC content, in particular `Xw` (e-mails, URLs), `Xe` (emoticons and emojis), `Xh` (hashtags, e.g. `#kvadogaja`) and `Xa` (mentions, e.g. `@dfiser3`).

The lemmatisation, which is also a part of the tool, takes into account the posited MSD and the lexicon; for pairs word-form : MSD that are already in the training lexicon it simply retrieves the lemma, while for others it uses its lemmatisation model.

[6] `http://nl.ijs.si/ME/V5/msd/`

3 The annotation campaign

A detailed overview of the sampling procedure, the annotation workflow and guidelines, and format conversions is given in [21]; here we briefly summarise these points.

The texts that constitute the manually annotated datasets were obtained by sampling the Janes corpus. In the initial stage two samples were made: Kons1, which includes tweets, and Kons2, which includes forum posts and comments on blog posts and news articles. Kons1 contained 4,000 tweets, which were sampled randomly but taking into account some constraints. First, we removed tweets longer than 120 characters, as these are often truncated, and tweets posted from corporate accounts, which typically do not display characteristics of CMC language. Furthermore, we wanted to have a sample containing both fairly standard language (so that we don't disregard standard but nevertheless CMC specific language) as well as very non-standard ones. We therefore took equal numbers (1,000) of T1L1, T3L1, T1L3 and T3L3 tweets. Likewise, Kons2 also contained 4,000 texts and was sampled according to the same criteria as Kons1. Since, unlike Twitter, these platforms do not impose a text length limit, we here took into account only texts between 20 and 280 characters in length in order to ensure a comparable sample in text length for Kons1 and Kons2.

Having correct tokenisation, sentence segmentation and normalisation was considered a priority, so Kons1 and Kons2 were first annotated for these levels. In the second phase, the already corrected subsets of the two datasets were reimported into the annotation tool as Kons1-MSD and Kons2-MSD and MSDs and lemmatisation were corrected manually. In the selected subsets for this second annotation campaign we preferred non-standard texts to standard ones, as we were aware that the dataset will be rather small and thus wanted to make it maximally CMC-specific.

Our Guidelines for CMC annotation mostly followed the Guidelines for annotating standard [12] and historical [4] Slovene texts but with some modifications regarding the differences of the medium (e.g. emoticons, URLs). At the normalisation level, special emphasis was given to the treatment of non-standard words with multiple spelling variants and without a standard form (e.g. `orng`, `ornk`, `oreng`, `orenk` for `'very'`), foreign language elements (e.g. `updateati`, `updajtati`, `updejtati`, `apdejtati` for `'to update'`) and linguistic features that are not normalised (e.g. hashtags, non-standard syntax and stylistic issues). At the lemmatisation and MSD levels, guidelines were designed to deal with foreign language elements, proper names and abbreviations as well as non-standard use of cases and particles.

The annotation was performed in WebAnno [22], a general-purpose web-based annotation tool that enables e.g. multi-layer annotation and features with multiple values. However, the tool is difficult to use for correcting tokenisation (and hence all the token dependent layers), so we had to introduce multivalued features and some special symbols in order to be able to split and merge tokens and assign sentence boundaries, as illustrated in Figure 1. Here the string `-3,8.` was wrongly treated as one word by the tokeniser, and the annotator corrected

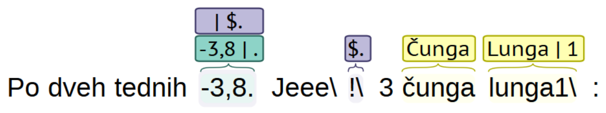


Fig. 1: Correcting token and sentence boundaries in WebAnno.

it to two tokens and inserted a sentence boundary after the second token (the full stop). It should also be noted that backslashes are used to indicate that the original text has no space between the tokens.

All the texts were first automatically annotated, then checked and corrected manually by a team of students. For the students a training and testing session was organised first. In the annotation campaigns, each text was annotated by two different annotators and then curated by the team leader.

We also put special emphasis on format conversion. The Janes corpus is encoded in TEI P5 [20], which WebAnno does not really support. We therefore developed a conversion from TEI to the WebAnno TSV tabular format, and a merge operation from the WebAnno exported TSV with the source TEI, resulting in a TEI encoding with corrected annotations. Given that we can change tokens in WebAnno this operation is fairly complex.

4 The Janes-Norm and Janes-Tag Datasets

As the end-result of the annotation we produced two datasets. Janes-Norm contains Kons1 and Kons2, i.e. it is meant as a gold-standard dataset for the annotation of tokenisation, sentence segmentation and normalisation. Janes-Tag is a subset of Janes-Norm and contains Kons1-MSD and Kons2-MSD, i.e. it is meant as a gold-standard dataset for the annotation of MSDs and lemmas. It should be noted that the order of the texts in both datasets was randomised so that it is easier to split them into training and testing sets while still retaining coverage over all text types.

4.1 Encoding and distribution

Both datasets are encoded in the same way. In particular, MSD tags and lemmas are also included in the Janes-Norm dataset, even though these were assigned automatically and thus contain errors. Nevertheless, even such annotations might prove useful for certain tasks, and it is easy enough to ignore or delete them if not needed.

Each dataset is encoded in XML as a TEI P5 [20] document, which includes its TEI header giving the metadata about the dataset and the body, which is composed of anonymous block elements (<ab>), each of which contains one text. Furthermore, each document also contains the MSD specifications encoded as a TEI feature-structure library. This makes it possible to decompose

```
<ab xml:id="janes.blog.publishwall.4264.3" type="blog" subtype="T1L3">
   <s>
      <w lemma="kaj" ana="#Rgp">Kaj</w><c> </c>
      <w lemma="biti" ana="#Va-r3s-y">ni</w><c> </c>
      <w lemma="ta" ana="#Pd-nsn">to</w><c> </c>
      <choice>
         <orig><w>tazadnje</w></orig>
         <reg>
            <w lemma="ta" ana="#Q">ta</w><c> </c>
            <w lemma="zadnji" ana="#Agpnsn">zadnje</w>
         </reg>
      </choice><c> </c>
      <choice>
         <orig><w>AAjevska</w></orig>
         <reg><w lemma="aa-jevski" ana="#Agpfsn">AA-jevska</w></reg>
      </choice><c> </c>
      <w lemma="molitev" ana="#Ncfsn">molitev</w>
      <pc ana="#Z">?</pc>
   </s>
</ab>
```

Fig. 2: TEI encoding of a text in the datasets.

an MSD into its individual features (attribute-value pairs) or localise it to Slovene.

As illustrated by Figure 2, each <ab> (i.e. a text) is labelled by its ID from the Janes corpus, its source (tweet, news, forum or blog) and its standardness score (T1L1, T1L3, T3L1 or T3L3) and then contains the contiguous sentences (<s>) containing the text. Tokens are encoded as words (<w>) or punctuation symbols (<pc>), and the original "linguistic" spacing is preserved in the TEI "character" element (<c>). Tokens are annotated with MSDs, which are pointers to their definition in the back-matter, with words also annotated with lemmas.

To encode the standard form of the words with non-standard orthography we use the TEI element <choice> with two subordinate elements, the original form(s) in <orig> and the normalised / regularised form(s) in <reg>. This complex approach has the advantage of allowing multiword mappings and distinguishing the annotation of the original from to that of the normalised form; as mentioned, we currently annotate only the normalised forms.

The TEI encoding was down-translated into the CQP vertical format used e.g. by Sketch Engine [19] and installed on the CLARIN.SI installation of noSketch Engine.

We did not perform any anonymisation on the datasets, as they are quite small and we thus do not consider them to pose a threat to privacy protection or actionable infringement of copyright or terms of use [7]. In the unlikely event

Table 1: Janes-Norm (sub)corpus sizes by standardness level and text type

	Texts		Tokens		Words		Norms		True norms		Multiw.	
All	7,816	100%	184,766	100%	143,929	100%	39,252	27.3%	16,498	42.0%	800	4.8%
T1L1	1,979	25.3%	48,438	26.2%	37,659	26.2%	7,878	20.9%	793	10.1%	78	9.8%
T1L3	1,936	24.8%	47,425	25.7%	35,569	24.7%	12,616	35.5%	6,548	51.9%	234	3.6%
T3L1	1,954	25.0%	41,474	22.4%	33,086	23.0%	6,457	19.5%	1,018	15.8%	153	15.0%
T3L3	1,947	24.9%	47,429	25.7%	37,615	26.1%	12,301	32.7%	8,139	66.2%	335	4.1%
blog	1,159	14.8%	20,987	11.4%	16,266	11.3%	3,566	21.9%	1,620	45.4%	88	5.4%
forum	1,572	20.1%	37,647	20.4%	31,002	21.5%	7,557	24.4%	3,789	50.1%	209	5.5%
news	1,145	14.6%	23,488	12.7%	19,160	13.3%	4,623	24.1%	1,876	40.6%	93	5.0%
tweet	3,940	50.4%	102,644	55.6%	77,501	53.8%	23,506	30.3%	9,213	39.2%	410	4.5%

of a complaint, we will remove the problematic text(s) from the public datasets on the concordancer(s).

We also plan to deposit Janes-Norm and Janes-Tag to the CLARIN.SI repository. Contrary to current practice in redistribution of CMC corpora (e.g. [9,2]) we will most likely distribute them under one of the CC licences.

4.2 Janes-Norm

The Janes-Norm dataset is meant for training and testing Slovene CMC tokenisers, sentence segmenters and word normalisation tools. Table 1 gives the size of the dataset overall and split into the included standardness levels and sources.

The complete dataset has 7,816 texts, which are, more or less, split equally among the four included standardness levels. The reason why the complete corpus does not have 8,000 texts and each split 2,000 texts is that the annotators had the option of marking individual texts as irrelevant (e.g. being completely in a foreign language), and these were then not included in the final dataset.

The texts contain almost 185.000 tokens or 144.000 words, where we count as a word all tokens except punctuation, numerals and tokens marked with one of the CMC-specific MSDs, i.e. emails, URLs, hashtags, mentions, emojis and emoticons. The table also shows that the proportions among the standardness levels are mostly preserved also in tokens and words. In terms of the text types, about half of the texts, tokens and words come from tweets, while about 15% of the texts are from blog and news comments each.

Moving to the number of words that have non-standard spelling, the "Norms" column shows the number of tokens that have been normalised, where the percentage is against the total number of words. "True norms" gives the number of linguistically more complex normalisations, i.e. where the normalisation goes beyond capitalisation or adding diacritics (e.g. `mačka` instead of `macka`) and the percentage refers to the number of normalised words. As can be seen, over a quarter (27.3%) of the words have been normalised, with 42% of these normalised at the morphological and lexical levels. Unsurprisingly, the

Table 2: Janes-Tag (sub)corpus sizes by standardness level and text type

	Texts		Tokens		Words		Norms		True norms		Multiw.	
All	2,958	100%	75,276	100%	56,562	100%	18,825	33.3%	10,102	53.7%	379	3.8%
T1L1	275	9.3%	6,695	8.9%	5,400	9.5%	954	17.7%	77	8.1%	11	14.3%
T1L3	1,219	41.2%	32,329	42.9%	23,159	40.9%	8,759	37.8%	4,447	50.8%	150	3.4%
T3L1	245	8.3%	4,559	6.1%	3,788	6.7%	589	15.5%	126	21.4%	12	9.5%
T3L3	1,219	41.2%	31,693	42.1%	24,215	42.8%	8,523	35.2%	5,452	64.0%	206	3.8%
blog	269	9.1%	5,046	6.7%	3,952	7.0%	848	21.5%	370	43.6%	24	6.5%
forum	403	13.6%	9,445	12.5%	7,761	13.7%	1,894	24.4%	934	49.3%	46	4.9%
news	303	10.2%	6,097	8.1%	4,801	8.5%	1,249	26.0%	522	41.8%	20	3.8%
tweet	1,983	67.0%	54,688	72.6%	40,048	70.8%	14,834	37.0%	8,276	55.8%	289	3.5%

more standard texts contain much less normalised words, with the L score significantly correlating with the need for normalisation. Looking at the text types, overall the most standard seem to be blog comments (21.9%), closely followed by forums and news comments, with tweets exhibiting the greatest proportion (30.3.%) of words requiring normalisation. The situation changes somewhat when we look at linguistically complex normalisations, as only 39.2% of tweets normalisations are the linguistically complex ones, followed by news (40.6%) and blog (45.4%) comments, and finally forums, where over half (50.1%) of the normalisations are linguistically complex, meaning that users take most care of diacritics and capitalisation on forums, and least on tweets, most likely stemming both from the instantaneous nature of the medium as well as typical input devices: forum posts on home computers vs. tweets on hand-held devices.

Finally, the last column shows the number and percentage against linguistic normalisations of cases where the normalisation involved splitting or joining words. As mentioned, these are especially difficult to model, so it is worth having a closer look at them. The results show that this is not a frequent phenomenon, involving only about 5% of linguistic normalisations. In other words, even if such cases are not treated at all, the overall drop in accuracy will not be very significant.

4.3 Janes-Tag

The Janes-Tag dataset is meant for training and testing Slovene CMC MSD taggers and lemmatisers. Similar to Janes-Norm, Table 2 gives the size of the dataset overall and split into the included standardness levels and sources.

The complete dataset has just under 3,000 texts and just over 75,000 tokens, giving over 56,000 words, i.e. it is about half the size of Janes-Norm. While this does not make for a large dataset (it is about one tenth of the size of ssj500k, the manually annotated corpus of standard Slovene) it is most likely enough to lead to significantly better Slovene CMC tagging and lemmatisation if tools were to be trained on a combination of ssj500k and Janes-Tag. Of course, it also gives us a gold-standard dataset for testing Slovene CMC taggers and lemmatisers.

Given the sampling criteria for Kons1-MSD and Kons2-MSD the proportions of texts, tokens and words among the standardness levels is quite different from Janes-Norm, as we here concentrated on L3 texts, which make up over 80% of the dataset. In terms of text types, the majority of the texts (67%) and even more of the tokens (72.8%) come from tweets, reflecting the dynamics of the annotation campaign. The normalisation-related percentages in the table are similar to those of Janes-Norm, probably varying due to mostly random sampling factors and are here included only for the sake of completeness.

5 Conclusions

In this paper we presented two manually annotated corpora meant for training and testing tools for tokenisation, sentence segmentation, word normalisation, morphosyntactic tagging and lemmatisation of Slovene CMC. We plan to use them to improve the accuracy of tools that we have developed for these tasks, which will then be used to re-annotate the complete Janes corpus. We have also made the datasets publicly available via the concordancer and plan to make it openly available for download as well, which is highly valuable for linguistic research as no such data has so far been made available for the analysis of non-standard Slovene. In addition, it will help other researchers or companies to improve or develop their own systems for analysing this increasingly important segment of the Slovene language.

The words in the datasets have been normalised to their standard spelling but there is currently no typology of the normalisations in the dataset. And while certain types of normalisation can be easily inferred automatically (diacritisation, capitalisation, word boundaries) others cannot. In particular, phonetic spelling cannot be automatially distinguished from typos, nor can combinations of normalisation types, such as missing diacritics + phonetic spelling be recognised. This information could be useful for linguistic investigations as well as for the profiling of normalisation tools, which is why we are considering launching another annotation campaign to add this information to the normalised words in the datasets in the near future. In addition, we would find it interesting to further investigate the multiword mappings from a linguistic and technical perspectives.

Acknowledgments. The authors would like to thank Kaja Dobrovoljc, Simon Krek, and Katja Zupan for their valuable contributions to the annotation guidelines, as well as the annotators who participated in this project: Teja Goli, Melanija Kožar, Vesna Koželj, Polona Logar, Klara Lubej, Dafne Marko, Barbara Omahen, Eneja Osrajnik, Predrag Petrović, Polona Polc, Aleksandra Rajković, and Iza Škrjanec. The work described in this paper was funded by the Slovenian Research Agency within the national research project "Resources, Tools and Methods for the Research of Nonstandard Internet Slovene" and the national research programme "Knowledge Technologies", by the Ministry of Education, Science and Sport within the "CLARIN.SI" research infrastructure, and by the Swiss National Science Foundation within the "Regional Linguistic Data Initiative" SCOPES programme.

References

1. Bartz, T., Beißwenger, M., Storrer, A.: Optimierung des Stuttgart-Tübingen-Tagset für die linguistische Annotation von Korpora zur internetbasierten Kommunikation: Phänomene, Herausforderungen, Erweiterungsvorschläge. Journal for Language Technology and Computational Linguistics 28(1), 157–198 (2014)
2. Chiari, I., Canzonetti, A.: Le forme della comunicazione mediata dal computer: generi, tipi e standard di annotazione. In: Garavelli, E., Suomela-Härmä, E. (eds.) Dal manoscritto al web: canali e modalità di trasmissione dell'italiano. Tecniche, materiali e usi nella storia della lingua. Franco Cesati Editore, Firenze
3. Dobrovoljc, K., Krek, S., Holozan, P., Erjavec, T., Romih, M.: Morphological lexicon Sloleks 1.2. Slovenian language resource repository CLARIN.SI (2015), `http://hdl.handle.net/11356/1039`
4. Erjavec, T.: The IMP historical Slovene language resources. Language Resources and Evaluation pp. 1–23 (2015)
5. Erjavec, T., Ignat, C., Poliquen, B., Steinberger, R.: Massive Multilingual Corpus Compilation: Acquis Communautaire and ToTaLe. In: The 2nd Language & Technology Conference - Human Language Technologies as a Challenge for Computer Science and Linguistics. Association for Computing Machinery (ACM) and UAM Fundacja (2005)
6. Erjavec, T.: MULTEXT-East: morphosyntactic resources for Central and Eastern European languages. Language Resources and Evaluation 46(1), 131–142 (2012), `http://dx.doi.org/10.1007/s10579-011-9174-8`
7. Erjavec, T., Čibej, J., Fišer, D.: Omogočanje dostopa do korpusov slovenskih spletnih besedil v luči pravnih omejitev (Overcoming Legal Limitations in Disseminating Slovene Web Corpora). Slovenščina 2.0: Empirical, Applied and Interdisciplinary Research 4(2), 189–219 (2016), `http://dx.doi.org/10.4312/slo2.0.2016.2.189-219`
8. Fišer, D., Erjavec, T., Ljubešić, N.: JANES v0.4: Korpus slovenskih spletnih uporabniških vsebin (Janes v0.4: Corpus of Slovene User Generated Content). Slovenščina 2.0: Empirical, Applied and Interdisciplinary Research 4(2), 67–99 (2016), `http://dx.doi.org/10.4312/slo2.0.2016.2.67-99`
9. Frey, J.C., Glaznieks, A., Stemle, E.W.: The DiDi Corpus of South Tyrolean CMC Data. In: Proceedings of the 2nd Workshop on Natural Language Processing for Computer-Mediated Communication / Social Media at GSCL2015 (NLP4CMC2015) (2015)
10. Gimpel, K., Schneider, N., O'Connor, B., Das, D., Mills, D., Eisenstein, J., Heilman, M., Yogatama, D., Flanigan, J., Smith, N.A.: Part-of-speech tagging for twitter: Annotation, features, and experiments. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2. pp. 42–47. HLT '11, Association for Computational Linguistics, Stroudsburg, PA, USA (2011), `http://dl.acm.org/citation.cfm?id=2002736.2002747`

11. Grčar, M., Krek, S., Dobrovoljc, K.: Obeliks: statistični oblikoskladenjski označevalnik in lematizator za slovenski jezik (obeliks: a statistical morphosyntactic tagger and lemmatiser for slovene). In: Zbornik Osme konference Jezikovne tehnologije. Ljubljana, Slovenia (2012)
12. Holozan, P., Krek, S., Pivec, M., Rigač, S., Rozman, S., Velušček, A.: Specifikacije za učni korpus. Projekt "Sporazumevanje v slovenskem jeziku" (Specifications for the Training Corpus. The "Communication in Slovene" project). Tech. rep. (2008), `http://www.slovenscina.eu/Vsebine/Sl/Kazalniki/K2.aspx`
13. Krek, S., Erjavec, T., Dobrovoljc, K., Može, S., Ledinek, N., Holz, N.: Training corpus ssj500k 1.3. Slovenian language resource repository CLARIN.SI (2013), `http://hdl.handle.net/11356/1029`
14. Ljubešić, N., Erjavec, T., Fišer, D.: Standardizing tweets with character-level machine translation. In: Proceedings of CICLing 2014. pp. 164–75. Lecture notes in computer science, Springer, Kathmandu, Nepal (2014)
15. Ljubešić, N., Fišer, D., Erjavec, T., Čibej, J., Marko, D., Pollak, S., Škrjanec, I.: Predicting the Level of Text Standardness in User-generated Content. In: Proceedings of Recent Advances in Natural Language Processing (2015)
16. Ljubešić, N., Erjavec, T.: Corpus vs. lexicon supervision in morphosyntactic tagging: the case of slovene. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016). European Language Resources Association (ELRA), Paris, France (may 2016)
17. Ljubešić, N., Erjavec, T., Fišer, D.: Corpus-based diacritic restoration for south slavic languages. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016). European Language Resources Association (ELRA) (may 2016)
18. Logar Berginc, N., Grčar, M., Brakus, M., Erjavec, T., Arhar Holdt, Špela., Krek, S.: Korpusi slovenskega jezika Gigafida, KRES, ccGigafida in ccKRES: gradnja, vsebina, uporaba (The Gigafida, KRES, ccGigafida and ccKRES corpora of Slovene language: compilation, content, use). Zbirka Sporazumevanje, Trojina, zavod za uporabno slovenistiko: Fakulteta za družbene vede, Ljubljana, Slovenia (2012)
19. Rychlý, P.: Manatee/Bonito - A Modular Corpus Manager. In: 1st Workshop on Recent Advances in Slavonic Natural Language Processing. pp. 65–70. Masarykova univerzita, Brno (2007)
20. TEI Consortium (ed.): TEI P5: Guidelines for Electronic Text Encoding and Interchange.
21. Čibej, J., Špela Arhar Holdt, Erjavec, T., Fišer, D.: Razvoj učne množice za izboljšano označevanje spletnih besedil (The Developoment of a Training Dataset for Better Annotation of Web Texts). In: Erjavec, T., Fišer, D. (eds.) Proceedings of the Conference on Language Technologies and Digital Humanities. pp. 40–46. Academic Publishing Division of the Faculty of Arts, Ljubljana, Slovenia (2016), `http://www.sdjt.si/jtdh-2016/en/`
22. Yimam, S.M., Gurevych, I., de Castilho, R.E., Biemann, C.: Webanno: A flexible,web-based and visually supported system for distributed annotations. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (System Demonstrations) (ACL 2013). pp. 1–6. Association for Computational Linguistics, Stroudsburg, PA, USA (Aug 2013)

Part II

Semantics and Language Modelling

Detecting Semantic Shifts in Slovene Twitterese

Darja Fišer[1,2] and Nikola Ljubešić[2,3]

[1] Dept. of Translation, Faculty of Arts, University of Ljubljana,
Aškerčeva cesta 2, SI-1000 Ljubljana, Slovenia
[2] Department of Knowledge Technologies, Jožef Stefan Institute,
Jamova cesta 3, SI-1000 Ljubljana, Slovenia
[3] Department of Information and Communication Sciences, University of Zagreb
Ivana Lučića 3, HR-10000 Zagreb, Croatia
darja.fiser@ff.uni-lj.si, nikola.ljubesic@ijs.si

Abstract. This paper presents first results of automatic semantic shift detection in Slovene tweets. We use word embeddings to compare the semantic behaviour of common words frequently occurring in a reference corpus of Slovene with their behaviour on Twitter. Words with the highest model distance between the corpora are considered as semantic shift candidates. They are manually analysed and classified in order to evaluate the proposed approach as well as to gain a better qualitative understanding of the nature of the problem. Apart from the noise due to preprocessing errors (45%), the approach yields a lot of valuable candidates, especially the novel senses occurring due to daily events and the ones produced in informal communication settings.

Key words: semantic shift detection, distributional semantics, word embeddings, user-generated content, tweets

1 Introduction

Meanings of words are not fixed but undergo changes, either due to the advent of new word senses or due to established word senses taking new shades of meaning or becoming obsolete (Mitra et al. 2015). These semantic shifts typically occur systematically (Campbell 2004), resulting in a meaning of a word to either expand/become more generalized, narrow down to include fewer referents or shift/transfer to include a new set of referents (Sagi et al. 2009). A classic example of expansion is the noun `miška/mouse` which used to refer to the small rodent but is now also used for describing the computer pointing device. The reverse process occurred with the noun `faks/faxs` that used to mean both the machine for telephonic transmission of printed documents and higher education institution, only the latter of which continues to be of use in contemporary colloquial Slovene.

There are also many cases in which words acquire new positive or negative connotations, processes that lexical semanticists call amelioration and pejoration (Cook and Stevenson 2009). Amelioration, which is especially frequent in slang, can be observed in the use of the adverb `hudo/terrific` which has a strong negative connotation in standard Slovene but has acquired a distinctly positive one in colloquial Slovene. Pejoration, the opposite effect of semantic shifts, can be observed in the use of the noun for `blondinka/blond woman`, which is neutral in standard Slovene but used distinctly pejoratively in informal settings.

2 Related work

While automatic discovery of word senses has been studied extensively Spark-Jones 1986; Ide and Veronis 1998; Schütze 1998; Navigli 2009), changes in the range of meanings expressed by a word have received much less attention, despite the fact that it is a very important challenge in lexicography where it is needed to keep the description of dictionary entries up-to-date. Apart from lexicography, up-to-date semantic inventories are also required for a wide range of human-language technologies, such as question-answering and machine translation. As more and more diachronic, genre- and domain-specific corpora are becoming available, it is becoming an increasingly attainable goal.

Most work in semantic shift detection focuses on diachronic changes in word usage and meaning by utilizing large historical corpora spanning several decades or even centuries (Mitra et al. 2015, Tahmasebi, Risse and Dietze 2011). Since we wish to look at the differences between standard and non-standard Slovene, our work is closer to the approaches conducted over two time points or corpora. Cook et al. (2013) induce word senses and identify novel senses by comparing the new 'focus corpus' with the 'reference corpus' using topic modelling for word sense induction. We instead chose to follow a simpler and potentially more robust approach which does not require us to discriminate specific senses, but which simply relies on measuring contextual difference of a lexeme in two corpora. In this respect, our work is similar to Gulordava and Baroni (2011) who detect semantic change based on distributional similarity between word vectors.

3 Data

In this paper we investigate semantic shifts in the 100-million token corpus of Slovene tweets (Fišer et al. 2016) with respect to the 1-billion token reference corpus Gigafida (Logar et al. 2012). We believe that user-generated content is an ideal resource to detect semantic shifts due to its increasing popularity and heterogeneous use(r)s, the language of which is all the more valuable because it is not covered by any of the existing traditional authoritative lexical and language resources.

We define headwords as lowercased lemmata expanded with the first two characters of the morphosyntactic description. The list of headwords of interest is produced by identifying lemmata with over 500 occurrences in our non-standard dataset that are also covered in the Sloleks lexicon[4] and are either common nouns (`Nc`), general adjectives (`Ag`), adverbs (`Rg`) or main verbs (`Vm`). Thereby we produced a list of 5425 lemmas.

4 Method

In this paper we test the suitability of using word embeddings to identify semantic shifts in user-generated content. This is a simple approach that relies on the basic principles of distributional semantics suggesting that one can model the meaning of a word by observing the contexts in which it appears (Firth 1957). Vector models position words in a semantic space given the contexts in which the words appear, making it possible to measure the semantic similarity of words as the distance between the positions in the semantic space, with CBOW and skip-gram (Mikolov et al., 2013) being nowadays the most widely used models.

We want to build two distributional models for each headword, one representing the headword in the standard language (from the Gigafida reference corpus), the other in non-standard language (from the Janes Twitter corpus).

Learning sparse representations of same words from different corpora is a straightforward task as these representations require context features to be counted and potentially processed with a statistic of choice. On the other hand, dense representations are based on representing each word in a way that maximises the predictability of a word given its context or vice versa. Given that the representation depends on the data available in each of the corpora, the representation learning for both corpora has to be performed in a single process. To do that, a trick has to be applied: encoding whether an occurrence of a headword came from the standard or non-standard dataset in form of a prefix to the headword itself (like `s_miška#Nc` for the occurrence in standard data and `n_miška#Nc` for the occurrence in non-standard data). Therefore the representation cannot be learned from running text as headwords need to have corpus information encoded while their contexts have to be free of that information so that they are shared between the two corpora.

The only tool that we know to accept already prepared pairs of headwords and context features is word2vecf[5]. Other tools accept running text only, limiting thereby the headwords and context features to the same phenomena like surface forms or lemmata.

As context features we use surface forms, avoiding thereby the significant noise introduced while tagging and lemmatising non-standard texts. The features are taken from a punctuation-free window of two words to each side of the headword. The relative position of each feature to the headword is not encoded. By following the described method, we produced dense vector

[4] `https://www.clarin.si/repository/xmlui/handle/11356/1039`
[5] `https://bitbucket.org/yoavgo/word2vecf`

representations of 200 dimensions for each of the 5425 lemmas for each of the two corpora.

We calculate the semantic shift simply as a cosine similarity, transformed to a distance measure, between the dense representation of a word built from standard and from non-standard data. More formally, for each $w \in V$ where w is a word and V is our vocabulary, we calculate the semantic shift of a word $ss(w)$ as

$$ss(w) = 1 - cossim(\boldsymbol{w_s}, \boldsymbol{w_n})$$

where the *cossim* function calculates the cosine similarity of two vectors, $\boldsymbol{w_s}$ is the 200-dimensional representation of the word calculated on the standard corpus data, and $\boldsymbol{w_n}$ the same representation on the non-standard corpus data.

5 Linguistic analysis of the results

We performed linguistic analysis on the top-ranking 200 lemmas from the reference and the Twitter corpus which display the most differences in their contexts. 90 (45%) of these were preprocessing errors in either corpus due to language identification, tokenisation, normalisation, tagging or lemmatisation errors (e.g. `talka/female hostage` which was a wrongly assigned lemma to the English word `talk`) and were therefore excluded from further analysis. This level of noise is not surprising as we are dealing with highly non-standard data that is difficult to process with high accuracy. At the same time, our analysis shows that this type of noise is highest at the top of the list and steadily decreases.

A detailed comparative analysis of the remaining 110 lemmas was performed by comparing Word Sketches of the same lemma in both corpora in the Sketch Engine concordancer (Kilgarriff et. al. 2014). The analysis of semantic shifts was performed in three steps. First, we tried to determine whether any semantic shift can indeed be detected. If yes, we further tried to determine whether the shift is minor or major. Finally, they were then classified into three subcategories each that are described in detail in the following section.

5.1 Minor semantic shifts

As the first type of minor shifts we considered those cases in which we identified the same senses in both corpora but with a different frequency distribution (e.g. `odklop`, which predominantly refers to the disconnecting of electricity, the internet etc. in the reference corpus but is most often used metaphorically in the Twitter corpus in the sense of taking a break, going on holiday or off-line to relax from work and every-day routine or `sesalec`, the predominant sense of which in Gigafida is mammal but vacuum cleaner in the Janes corpus).

Second, we also counted the cases in which distinct discrepancies were detected in the patterns in which a word is regularly used, influencing the sense of the target word (e.g. `kvadrat/square`, which is almost exclusively used in the pattern `na kvadrat/squared` on Twitter, or `eter/ether`, which on Twitter is almost exclusively used in the pattern `v eter/on air`).

The third type of minor shifts we detected is the narrowing of a word's semantic repository that is most likely not a sign of a word sense dying out but rather due to a limited set of topics present in Twitter discussions with respect to the set of topics in the reference corpus (e.g. `posodobiti`, which is only used in the IT sense on Twitter, never as `modernise` in general as is frequent in Gigafida, or `podnapis`, with which Twitter users only referes to subtitles, never to captions below images etc. as is frequent in Gigafida).

5.2 Major semantic shifts

As the first type of major shifts we considered novel usage of words that is a direct consequence of daily events, political situations, natural disasters or social circumstances (e.g. `vztrajnik` which traditionally meant flywheel but started being used to refer to the persistent protesters in the period of political and social unrest in 2012-2013 or `pirat` who used to be confined to the sea but can now be found on the internet as well and even in politics as members of the new party, only the latter in distinctly positive contexts). It would be interesting to track whether such semantic shifts are short-lived or which of them become a permanent part of our lexico-semantic repository.

Second, many new senses in the Twitter corpus can be detected because a lot of informal communication is performed via Twitter and colloquial language is frequent (e.g. `optika` which is used to refer to the lense mechanism in different devices, a store that sells glasses or a viewpoint in standard Slovene but is often used to refer to broadband internet in informal settings, or `carski` which is an adjective to refer to emperor but is used as a synonym of great, wonderful in non-standard language).

Finally, some new communication conventions have emerged on social media which resulted in some novel word senses as well (e.g. `sledilec`, a person who follows you on Twitter or other social media, or `opomnik`, a reminder message on the computer or telephone).

5.3 Results and discussion

As can be seen in Table 1, some type of semantic shift was detected in 75% of the cases in the sample that was analysed, suggesting the proposed approach to be quite accurate, given the complexity of the task. A large majority of all the semantic shifts detected were major shifts (74% of all the shifts detected). Unsurprisingly, most semantic shifts can be attributed to discussing daily events and to using informal, colloquial language on Twitter. At the same time, these are also the most interesting cases from the research perspective because they are still missing in all the available lexico-semantic resources of Slovene, which proves the suitability of the proposed approach for the task. In addition, some highly creative attribution of new meaning to common words has also been detected (e.g. `kahla/potty` which refers to a politician Karel Erjavec who cannot pronounce letter `r`, or `pingvin/penguin`, a derogatory nickname of the leader of a political party), showing that Twitter users play with language skilfully and are quick to adopt new coinages. The results of the performed linguistic analysis thus show that the approach presented in this paper could significantly contribute to regular semi-automatic updates of corpus-based general as well as specialized dictionaries.

Table 1: Types of semantic shifts in Slovene tweets

	No.	%
No shift	28	25%
Minor shift	21	19%
Semantic narrowing	3	3%
Usage pattern	6	5%
Redistribution of senses	12	11%
Major shift	61	56%
CMC-specific	6	5%
Colloquial	23	21%
Events	32	29%

The detected minor shifts systematically show the differences in the focus and range of topics between the two corpora. The fact that many more novel usages than narrowings were detected suggests that the reference corpus could be further enhanced with texts from social media and other less formal and standard communication practices as they contain rich and valuable linguistic material that is now absent in the reference corpus.

6 Conclusion

In this paper we presented the first results of automatic semantic shift detection for the Slovene used in social media. We measured the semantic shift of a word as the distance between the word embedding representation learned from a reference corpus of Slovene and the word embedding learned on a Twitter corpus of Slovene. We performed a manual analysis of 200 words with the highest measurements. The analysis shows that apart from the noise due to preprocessing errors (45%) that are easy to spot, the approach yields a lot of highly valuable semantic shift candidates, especially the novel senses occurring due to daily events and the ones produced in informal communication settings. The results of this experiment will be used in the development of the dictionary of Slovene Twitterese (Gantar et al. 2016).

Our future work will focus on (1) extending the manual analysis to lower-ranked candidates, (2) extending the approach to lower-frequency candidates, (3) comparing our method with alternative methods such as representing words as word sketches / syntactic patterns and (4) using supervised learning for detecting semantic shifts, discriminating between specific types of semantic shifts or filtering preprocessing errors.

Acknowledgments. The work described in this paper was funded by the Slovenian Research Agency within the national basic research project "Resources, Tools and Methods for the Research of Nonstandard Internet Slovene" (J6-6842, 2014-2017) and the COST Action "European Network of e-Lexicography" (IS1305, 2013-2017).

References

1. Blei, DM.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet Allocation. The Journal of Machine Learning Research 3: 993 – 1022 (2003).
2. Campbell, L. Historical linguistics: An introduction. Cambridge, MA: The MIT Press. (2004)
3. Cook, P., Stevenson, S. CALC '09 Proceedings of the Workshop on Computational Approaches to Linguistic Creativity, pp. 71–78 (2009).
4. Cook, P., Lau, J. H., Rundell, M., McCarthy, D., Baldwin, T.: A lexicographic appraisal of an automatic approach for detecting new word senses. In Electronic lexicography in the 21st century: thinking outside the paper. Proceedings of the eLex conference, Tallinn, Estonia, 49–65 (2013).
5. Firth, J.R.: A Synopsis of Linguistic Theory. Studies in Linguistic Analysis. (1957).
6. Fišer, D., Erjavec, T., Ljubešić, N.: JANES v0.4: Korpus slovenskih spletnih uporabniških vsebin Slovenščina 2.0 4/2, 67–99 (2016).
7. Gantar P., Škrjanec I., Fišer D., Erjavec T.: Slovar tviterščine. Proceedings of the Conference on Language Technologies and Digital Humanities, Ljubljana, Slovenia, 71–76. (2016)
8. Gulordava, K., Baroni, M.: A distributional similarity approach to the detection of semantic change in the Google Books Ngram corpus. In Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics, 67–71. (2011)
9. Ide, N., Véronis, J.: Introduction to the special issue on word sense disambiguation: the state of the art. Computational linguistics 24/1, 2-40. (1998)
10. Kilgarriff, A., et al.: The Sketch Engine: ten years on. In Lexicography 1—30, (2014).
11. Logar Berginc, N., Grčar, M., Erjavec, T., Arhar Holdt, Š., Krek, S.: Korpusi slovenskega jezika Gigafida, KRES, ccGigafida in ccKRES: gradnja, vsebina, uporaba. Trojina, zavod za uporabno slovenistiko: Fakulteta za družbene vede. Zbirka Sporazumevanje, Ljubljana, Slovenia. (2012)
12. Mikolov, T., Yih, W., Zweig, G.: Linguistic Regularities in Continuous Space Word Representations. HLT-NAACL, 746—751. (2013)
13. Mitchell, T.M.: Machine Learning, McGraw-Hill, Inc. New York, NY, USA. (1997)
14. Mitra, S., et al.: An automatic approach to identify word sense changes in text media across timescales. Natural Language Engineering 21/05, 773–798. (2015)
15. Navigli, R.: Word sense disambiguation: A survey. ACM Computing Surveys (CSUR) 41/2. (2009)

16. Sagi, E., Kaufmann, S., Clark, B.: Semantic Density Analysis: Comparing Word Meaning across Time and Phonetic Space. In Proceedings of the EACL 2009 Workshop on GEMS: GEometical Models of Natural Language Semantics, pages 104-111. Athens, Greece. (2009)
17. Schütze, H.: Automatic word sense discrimination. Computational linguistics 24/1, 97–123. (1998)
18. Spark-Jones, K.: Synonym and Semantic Classification. Edinburgh Information Technology Series. Edinburgh University Press, Edinburgh. (1986)
19. Tahmasebi, N., Risse, T., Dietze, S.: Towards automatic language evolution tracking, a study on word sense tracking. In Joint Workshop on Knowledge Evolution and Ontology Dynamics. (2011)

The Algorithm of Context Recognition in TIL

Marie Duží, Michal Fait

VSB-Technical University Ostrava, Department of Computer Science FEI,
17. listopadu 15, 708 33 Ostrava, Czech Republic
marie.duzi@vsb.cz, michal.fait@vsb.cz

Abstract. The goal of this paper is to introduce the algorithm of context recognition in the functional programming language *TIL-Script*. The TIL-Script language is an operationally isomorphic syntactic variant of Tichý's Transparent Intensional Logic (TIL). From the formal point of view, TIL is a hyperintensional, partial, λ-calculus with *procedural* semantics. Due to ramified hierarchy of types it is possible to distinguish three levels of abstraction at which TIL constructions operate. At the highest *hyperintensional* level the object to operate on is a *construction* (though a higher-order construction is needed to present this lower-order construction as an object of predication). At the middle *intensional* level the object to operate on is the *function* presented, or constructed, by a construction, while at the lowest *extensional* level the object to operate on is the *value* (if any) of the presented function. Thus, a necessary condition for the development of an inference machine for the *TIL-Script* language is recognizing a context in which a construction occurs, namely extensional, intensional and hyperintensional context, so that inference rules can be properly applied.

Key words: Transparent Intensional Logic, TIL-Script, three kinds of context, context-recognition algorithm

1 Introduction

The family of automatic theorem provers, known today as HOL, is getting increasingly interest in logic, mathematics, and computer science.[1] These tools are broadly used in automatic theorem checking and applied as interactive proof assistants. As 'HOL' is an acronym for higher-order logic, the underlying logic is usually a version of a simply typed λ-calculus. This makes it possible to operate both in extensional and intensional contexts, where a value of the denoted function or the function itself, respectively, is an object of predication.

Yet there is another application that is gaining interest, namely natural-language processing. There are large amount of text data that we need to analyse and formalize. Not only that, we also want to have question-answer systems which would infer implicit computable knowledge from these large explicit knowledge bases. To this end not only intensional but rather hyperintensional logic is needed, because we need to formally analyse natural language

[1] See, for instance, [1] or [6].

in a fine-grained way so that the underlying inference machine is neither over-inferring (that yields inconsistencies) nor under-inferring (that causes lack of knowledge). We need to properly analyse agents' attitudes like knowing, believing, seeking, solving, designing, etc., because attitudinal sentences are part and parcel of our everyday vernacular. And attitudinal sentences, *inter alia*, call for a hyperintensional analysis, because substitution of a logically equivalent clause for what is believed, known, etc. may fail. Hyperintensional individuation is frequently also referred to as 'fine-grained' or sometimes simply 'intensional' individuation, when 'intensional' is not understood in the specific sense of possible-world semantics or in the pejorative sense of flouting various logical rules of extensional logic.

A principle of individuation qualifies as hyperintensional as soon as it is finer than necessary equivalence. The main reason for introducing hyperintensionality was originally to block various inferences that were argued to be invalid. The theoretician introduces a notion of hyperintensional context, in which the proper substituends are hyperintensions rather than the modal intensions of possible-world semantics or extensions. For instance, if Tilman is solving the equation $sin(x) = 0$, then he is not solving the infinite set of multiples of the number π, because this set is the solution and there would be nothing to solve, the solver would immediately be a finder. Yet there is something Tilman is solving. He is trying to execute the procedure specified by $\lambda x.sin(x)=0$. Thus, there is the other side of the coin, which is the positive topic of which inferences should be validated in hyperintensional contexts.

TIL definition of hyperintensionality is positive rather than negative. Any context in which the meaning of an expression is *displayed* rather than *executed* is hyperintensional.[2] Moreover, our conception of meaning is *procedural*. Hyperintensions are abstract procedures rigorously defined as TIL constructions which are assigned to expressions as their context-invariant meanings. This entirely anti-contextual and compositional semantics is, to the best of our knowledge, the only one that deals with all kinds of context, whether extensional, intensional or hyperintensional, in a uniform way. The same extensional logical laws are valid invariably in all kinds of context. In particular, there is no reason why Leibniz's law of substitution of identicals, and the rule of existential generalisation were not valid. What differ per the context are not the rules themselves but the types of objects on which these rules are applicable. In an *extensional* context they are *values* of the functions denoted by the respective expressions; in an *intensional context* the rules are applicable on the denoted *functions* themselves, and finally in a *hyperintensional context* the *procedures* that is the meanings themselves are the objects to operate on. Due to its stratified ontology of entities organised in a ramified hierarchy of types, TIL is a logical

[2] In [2] we use the terms 'mentioned' vs. 'used'. But since these terms are usually understood linguistically as using and mentioning *expressions*, whereas in TIL we use or mention their *meanings*, here we vote for 'displayed' vs. 'executed', respectively. See also [3].

framework within which such an extensional logic of hyperintensions has been introduced.[3]

The *TIL-Script* language is an operationally isomorphic syntactic variant of TIL. The development of its inference machine is based on these principles. First, we implement the algorithm of *context recognition* that makes it possible to determine the type of an object to operate on. Second, we implement TIL *substitution method* (including β-conversion 'by value') that makes it possible to operate on displayed constructions in a hyperintensional context. Finally, we are going to implement a *hyperintensional variant of the sequent calculus* for TIL that has been specified in [4] and [8].

The goal of this paper is the description of the first step, that is of the context-recognition algorithm. The rest of the paper is organised as follows. In Section 2 we introduce basic principles of the TIL-Script language. The algorithm of context recognition is described in Section 3 and concluding remarks can be found in Section 4.

2 Basic Principles of TIL and the TIL-Script language

The TIL syntax will be familiar to those who are familiar with the syntax of λ-calculi with four important exceptions. *First*, TIL λ-terms denote abstract procedures rigorously defined as *constructions*, rather than the set-theoretic functions produced by these procedures.[4] Thus the construction *Composition* symbolised by $[FA_1 \dots A_m]$ is the very procedure of applying a function presented by F to an argument presented by $A_1, \dots, A_m$, and the construction *Closure* $[\lambda x_1 x_2 \dots x_n C]$ is the very procedure of constructing a function by λ-abstraction in the ordinary manner of λ-calculi. *Second*, objects to be operated on by complex constructions must be supplied by atomic constructions. Atomic constructions are one-step procedures that do not contain any other constituents but themselves. They are *variables* and *Trivialization*. Variables construct entities of the respective types dependently on valuation, they v-construct. For each type there are countably many variables assigned that range over this type (v-construct entities of this type). Trivialisation $'X$ of an entity X (of any type even a construction) constructs simply X. In order to operate on X, the object X must be grabbed first. Trivialisation is such a one-step grabbing mechanism. *Third*, since the product of a construction can be another construction, constructions can be executed twice over. To this end we have *Double Execution* of X, ^2X, that v-constructs what is v-constructed by the product of X. Finally, since we work with partial functions, constructions can be v-improper in the sense of failing to v-construct an object for a valuation v.[5]

[3] See, for instance,[4].

[4] For details see [9] and [3].

[5] TIL is one of the few logics that deal with partial functions, see also [7]. There are two basic sources of improperness. Either a construction is not type-theoretically coherent, or it is a procedure of applying a function f to an argument a such that f is not defined

Since TIL has become a well-known system, see, for instance [2], [9], and other numerous papers, in what follows we introduce only the grammar of the TIL-Script language, and characterize the syntactic differences between TIL and TIL-Script. The TIL-Script functional declarative language is a computational variant of TIL. It covers all the functionalities specified in the TIL system but slightly differs in its notation that applies purely the ASCII code. Thus, the syntax does not involve subscripts and superscripts, Greek characters, and special characters like '∀'or '∃'. These special symbols have been replaced by the key-words like 'Exist', 'ForAll', 'Bool', 'Time', 'World'. Greek 'λ' in Closure is replaced by '\'. The abbreviated '$\alpha_{\tau\omega}$' is in TIL-Script written as 'α@tw'. On the other hand, the set of basic data types is richer here to cover those useful in functional programming. In TIL-Script we distinguish the types of real and natural numbers from discrete times, and we also have the type String. Higher-order types $*_n$ are just $*$, we do not mark up the order of constructions, because it is controlled by the syntactic analyzer. In TIL-Script we also work with the functional type of a *List*. This type is defined by the key-word List and the types of its elements. For instance, List (Real) is a list of real numbers. Though this is a derived type, because each list can be defined as a function mapping natural numbers to the respective types, in practice it is much more convenient to work directly with the type List. The differences in basic types are summarized in Table 1.

Table 1: *TIL-Script* basic types

TIL	TIL-Script	*Description*
ο	Bool	*Truth-values*
ι	Indiv	*Individuals (universe)*
τ	Time	*Times*
ω	World	*Possible worlds*
-	Int	*Integers*
τ	Real	*Real numbers*
-	String	*String of characters*
α	Any	*Unspecified type*
$*_n$	$*$	*Constructions of order n*

Here is the TIL-Script grammar. It is easy to check that this grammar specifies the language functionally isomorphic to the language specified in the classical TIL.

```
start = {sentence};
sentence = sentence content , termination;
```

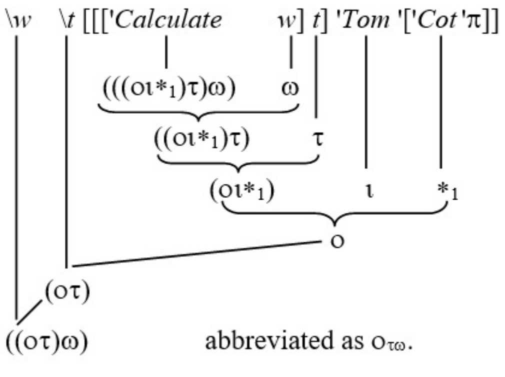
at a. For instance, Composition $[{}'Cotg\ {}'\pi]$ is v-improper for any valuation v, because the function cotangent is not defined at the number π in the domain of real numbers. *Single Execution* ^1X is improper for any valuation v in case X is not a construction.

```
sentence content = type definition | entity definition |
    construction | global variable definition;
termination = optional whitespace,".", optional whitespace;

type definition = "TypeDef", whitespace, type name, optional
    whitespace, ":=", optional whitespace, data type;
entity definition = entity name, {optional whitespace,",",
    optional whitespace, entity name}, optional whitespace,
    "/", optional whitespace, data type;
global variable definition = variable name, {optional
    whitespace, ",", optional whitespace, variable name},
    optional whitespace, "->", optional whitespace, datatype;
construction = (trivialisation | variable | composition |
    closure | n-execution) [, "@wt"];

data type = (embeded type | list type | touple type | user
    type | enclosed data type) [, '@tw'];
embeded type = "Bool" | "Indiv" | "Time" | "String" | "World"
     | "Real" | "Int" | "Any" | "*";
list type = "List", optional whitespace, "(", optional
    whitespace, data type, optional whitespace, ")";
touple type = "Tuple", optional whitespace, "(", optional
    whitespace, data type, optional whitespace, ")";
user type = type name;
enclosed data type = "(", optional whitespace, data type, {
    whitespace, data type}, optional whitespace ")";

variable = variable name;
trivialisation = "'", optional whitespace, (construction |
    entity);
composition = "[", optional whitespace, construction,
    optional whitespace, construction, {construction},
    optional whitespace, "]";
closure = "[", optional whitespace, lambda variables,
    optional whitespace, construction, optional whitespace,
    "]";
lambda variables = "\", optional whitespace, typed variables;
typed variables = typed variable, {optional whitespace,",",
    typed variable};
typed variable = variable name, [optional whitespace, ":",
    optional whitespace, data type];
n-execution = "^", optional whitespace, nonzero digit,
    optional whitespace, (construction | entity);

entity = keyword | entity name | number | symbol;

type name = upperletter name;
entity name = upperletter name;
variable name = lowerletter name;
```

```
keyword = "ForAll" | "Exist" | "Every" | "Some" | "True" | "
    False" | "And" | "Or" | "Not" | "Implies";
lowercase letter = "a" | "b" | ... | "z";
uppercase letter = "A" | "B" | ... | "Z";
symbols = "+" | "-" | "*" | "/";
zero = "0";
nonzero digit = "1" | "2" | ... | "9";
number = ( zero | nonzero digit),  { zero | nonzero digit }
    [".",  (zero | nonzero digit), { zero | nonzero digit }];
upperletter name = uppercase letter, { lowercase letter |
    uppercase letter | "_" | zero | nonzero digit };
lowerletter name = lowercase letter, { lowercase letter |
    uppercase letter | "_" | zero | nonzero digit };
whitespace = whitespace character, optional whitespace;
optional whitespace = { whitespace character };
whitespace character = ? space ? | ? tab ? | ? newline ?;
```

To illustrate TIL-Script analysis, we adduce an example of the analysis of the sentence "Tom calculates cotangent of the number π" followed by its derivation tree with type assignment. In the interest of better readability and a clear arrangement of the derivation tree, we use here Greek letters for types, as it is in classical TIL. According to the above grammar, the types are as follows: ο= Bool, ι= Indiv, τ= Time, ω= World, $\omicron_{\tau\omega}$ = Bool@tw, $*_n = *$.

$$\backslash w \backslash t[[['Calculate\ w]\ t]\ 'Tom\ '['Cot'\pi]]$$



The resulting type is $\omicron_{\tau\omega}$, that is the type of the proposition that Tom calculates Cotangent of π. The types of the objects constructed by 'π, '*Cot* and ['*Cot* 'π], that is τ, (ττ) and τ, respectively, are irrelevant here, because these constructions are not constituents of the whole construction. They occur only displayed by Trivialization '['*Cot* 'π], that is hyperintensionally. We are going to deal with this issue in the next section.

3 Context Recognition

The algorithm of context recognition is based on definitional rules presented in [2], §2.6. Since these definitions are rather complicated, here we introduce just the main principles. TIL operates with a fundamental dichotomy between hyperintensions (procedures) and their products, i.e. functions. This dichotomy corresponds to two fundamental ways in which a construction (meaning) can occur, to wit, *displayed* or *executed*. If the construction is displayed, then the procedure itself becomes an object of predication; we say that it occurs *hyperintensionally*. If the construction occurs in the execution mode, then it is a constituent of another procedure, and an additional distinction can be found at this level. The constituent presenting a function may occur either *intensionally* or *extensionally*. If intensionally, then the whole function is an object of predication; if extensionally, then a functional value is an object of predication. The two distinctions, between displayed/executed and intensional/extensional occurrence, enable us to distinguish between the three kinds of context:

– *hyperintensional context*: a construction occurs in a displayed mode (though another construction at least one order higher needs to be executed to produce the displayed construction)
– *intensional context*: a construction occurs in the executed mode to produce a function but not its value (moreover, the executed construction does not occur within another hyperintensional context)
– *extensional context*: a construction occurs in the executed mode in order to produce particular value of a function at a given argument (moreover, the executed construction does not occur within another intensional or hyperintensional context).

The basic idea underlying the above trifurcation is that the same set of logical rules applies to all three kinds of context, but these rules operate on different complements: procedures, produced functions, and functional values, respectively. A substitution is, of course, invalid if something coarser-grained is substituted for something finer-grained.

The algorithm of context recognition is realized in the Prolog programming language.[6] In the phase of the syntactic analysis of the TIL-Script language, a Prolog database of constructions and types is created. This database consists of three main parts:

1. *Typed objects* are pairs (name, type), represented as binary relations (type/2).
2. *Typed* global *variables* are pairs (name, type) represented as binary relations (globalVariable/2).
3. *Constructions*; constructions are the most complicated case. They are represented as 10-ary relations (construction/10);

[6] Context recognition system download is available here: `http://elogika.vsb.cz/projects/GA15-13277S/`.

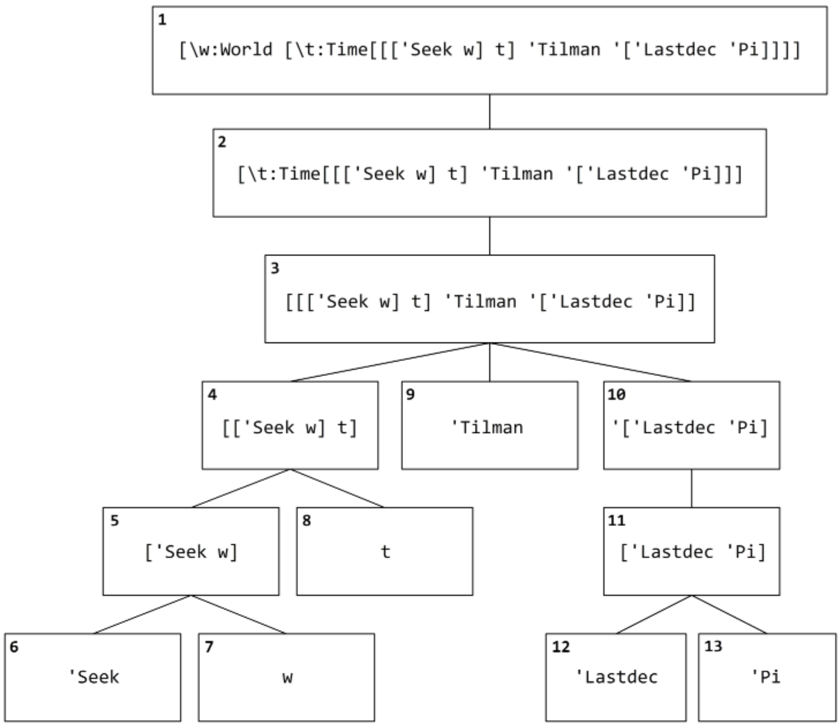

1
[\w:World [\t:Time[[['Seek w] t] 'Tilman '['Lastdec 'Pi]]]]
2
[\t:Time[[['Seek w] t] 'Tilman '['Lastdec 'Pi]]]
3
[[['Seek w] t] 'Tilman '['Lastdec 'Pi]]
4
[['Seek w] t]
9
'Tilman
10
'['Lastdec 'Pi]
5
['Seek w]
8
t
11
['Lastdec 'Pi]
6
'Seek
7
w
12
'Lastdec
13
'Pi

Fig. 1: Derivation tree of the construction

Each construction or subconstruction denoted by a term of an input TIL-Script file is recorded as a relation construction/10. For example, having the term '[\x ['+ '5 '4]]', five records of construction/10 are created, because five constructions are denoted here; they are [\x ['+ '5 '4]], ['+ '5 '4], '+, '5 and '4. Each record contains a unique identifier ID, which can be referred to by another construction. In this way we obtain a derivation tree structure of each construction. Figure 1 illustrates the tree structure of the construction assigned to the sentence "Tillman is seeking last decimal of the number π." As it is common in Prolog, the algorithm applies the depth-first search strategy with backtracking. The numbers of nodes in Figure 1 illustrate this strategy. The algorithm recursively calls itself for every child node or terminates if the current node is a leaf (that is a construction record without a child ID list). Whenever a leaf node is reached, backtracking comes into the scene and another branch is being searched.

The algorithm of context recognition consists of several steps. First, we recognize hyperintensional occurrences of displayed constructions, that is those that occur in the scope of a Trivialization, the effect of which, however, has not been cancelled by Double Execution, because ^{2}'C is equivalent to C. Moreover, a higher-level context is dominant over a lower-level one. Thus, if C occurs in D hyperintensionally, then all the subconstructions of C occur hyperintensionally in D as well. Here is the algorithm to determine hyperintensional occurrences.

```
Name: determineHyperintensional
Input: construction C, constant S indicating whether a current
construction occurrence is displyed or executed; in the beginning
S is set to unknown.

If S=''mentioned''
    Save hyperintensional context of construction C
If S=''used'' and C is trivialisation of another construction ‘D
    call determineHyperintensional(D,"mentioned")
else
    If S=''used'' and C is double execution ^2D and D is
    trivialization of a construction 'E
        call determineHyperintensional(E,"used")
    Else
        P = child nodes of construction C
        For every construction X in P do
            determineHyperintensional (X,S)
```

If the occurrence of C within D is not hyperintensional, then it occurs in the execution mode as a constituent of D, and the object that C v-constructs (if any) plays the role of an argument to operate on. In such a case we have to distinguish whether C occurs *intensionally* or *extensionally*. To this end we first distinguish between extensional and intensional supposition of C. Since C occurs executed, it is typed to v-constructs a function f of type $(\alpha\beta_1 \ldots \beta_n)$, n possibly equal to zero. Now C may be composed within D with constructions $D_1 \ldots D_n$ which are typed to v-construct the arguments of the function f, that is Composition $[CD_1 \ldots D_n]$ is a constituent of D. In such a case we say that C occurs in D with *extensional supposition*. Otherwise C occurs in D with *intensional supposition* that is intensionally.

The algorithm first determines occurrences with extensional supposition, and then it is in a position to check whether a given construction that occurs with extensional supposition is, or is not occurring within a λ-generic context. If the context is non-generic, then the respective occurrence is extensional. Otherwise, the context is intensional. Here is the algorithm for extensional-supposition recognition.

```
Name: extSupositionConstructions
Output: set of constructions with extensional suposition

For every construction C do
If C is composition [X Y1...Ym]
    If C does not occur in hyperintensional context
```

```
        Add construction X to the result
If C is execution ^1X or ^2X, where X is object of order one
    If C does not occur in hyperintensional context
        Add construction C to the result
For every execution C in form ^2X, where X v-constructs object of
order one
    If C does not occur in hyperintensional context
        Add construction C to the result
```

The algorithm checking λ-genericity is specified as per Def. 11.6 of [5]. In principle, it checks whether to each Closure there is a pairing Composition. This is so because the procedure of constructing a function by λ-abstraction raises the context up to the intensional level, while the dual procedure of applying a function to an argument decreases the context down. The algorithm examines the derivation tree of a construction and dynamically creates a generic-type list that determines the genericity level. If the list is empty, the context is non-generic, otherwise it is λ-generic.

For example, the generic-type list of the Trivialisation '+ occurring in the following constructions is as follows:

- in Composition ['+ x y] the list is empty, hence non-generic context;
- in Closure [\x:Real ['+ x y]] the context is [[Real]]-generic;
- in Closure [\y:Real [\x:Real ['+ x y]]] the context is [[Real],[Real]]-generic;
- in Composition [[\y:Real [\x:Real ['+ x y]]] '5] the context is again [[Real]]-generic;
- in Composition [[[\y:Real [\x:Real ['+ x y]]] '5] '5] the list is empty, hence non-generic context;
- in Closure [\x:Real, y:Real ['+ x y]] the context is [[Real,Real]]-generic.

The algorithm for generic type recognition is specified as follows:

```
Name: genericity
Input: constructions D and C, where D is constituent C
Output: generic type

1.If C is atomic construction and C = C
      result = non-generic type
2.if C is closure [\x1,...,xm X]; x1 $\to_v$ γ1, ..., xm $\to_v$ γm, then:
      a)If D = C,
             β = genericity(X,X)
             result = ((γ1,...,ym)β)
      b)If D is constituent X,
             β = genericity(D,X)
             result = ((y1,...,ym)β)
3.If C is composition [X Y1...Ym]; Y1 $\to_v$ γ1,...,Ym $\to_v$ γm.
      a)If D=C
             result = genericity(X,C)
```

```
     b)If D is constituent X
            G=genericity(X,X)
            If G is non-generic type
                result=genericity(D,X)
            Else if G is generic type  ((γ1,...,γm)β)
                result = β
     c)If D is constituent of one construction Y from Yi
               result = genericity(D,Y)
4.If C is execution ^2X or ^1X where X is construction
     If D is constituent X
         result = genericity(D,X)
```

The last step of the context-recognition algorithm is easy. For every construction, if the occurrence is not hyperintensional or extensional, the result is intensional context.

```
Name: determineIntensional
    For every construction C
        If C does not occur intensionally neither hyperintensionally
            Save intensional context of construction C
```

Here is an example of the result of the syntactic analysis including context-recognition of the construction

[\w: World [\t:Time ['Seek@wt 'Tilman '['Lastdec 'Pi]]]].

```
<construction occurrence ="Intensional"
              construction ="[\w:World [\t:Time [[['Seek w] t] 'Tilman '['Lastdec 'Pi]]]]">
  <construction occurrence ="Intensional" construction ="[\t:Time [[['Seek w] t] 'Tilman '['Lastdec 'Pi]]]">
    <construction occurrence ="Intensional" construction ="[[['Seek w] t] 'Tilman '['Lastdec 'Pi]]">
      <construction occurrence ="Intensional" construction ="[['Seek w] t]">
        <construction occurrence ="Intensional" construction ="['Seek w]">
          <construction occurrence ="Intensional" construction ="'Seek"></construction>
          <construction occurrence ="Intensional" construction ="w"></construction>
        </construction>
        <construction occurrence ="Intensional" construction ="t"></construction>
      </construction>
      <construction occurrence ="Intensional" construction ="'Tilman"></construction>
      <construction occurrence ="Intensional" construction ="'['Lastdec 'Pi]">
        <construction occurrence ="Hyperintensional" construction ="['Lastdec 'Pi]">
          <construction occurrence ="Hyperintensional" construction ="'Lastdec"></construction>
          <construction occurrence ="Hyperintensional" construction ="'Pi"></construction>
        </construction>
      </construction>
    </construction>
  </construction>
</construction>
```

4 Conclusion

We introduced the computational variant of Transparent Intensional Logic, the TIL-Script functional programming language. Our main novel result is the implementation of the algorithm that recognises three kinds of context, namely extensional, intensional and hyperintensional, which is a necessary condition for the implementation of the TIL-Script inference machine. It is an important result, because when testing the algorithm, it turned out that there are still slight unintended inaccuracies in the definitions as presented in [5], which in turn led to their revision.

Acknowledgments. This work has been supported by the Grant Agency of the Czech Republic project No. GA15-13277S, "Hyperintensional logic for natural language analysis" and by the internal grant agency of VSB-Technical University Ostrava, project No. SP2016/100, "Knowledge modelling and its applications in software engineering II".

References

1. Benzmüller, Ch. (2015): Higher-Order Automated Theorem Provers. In *All about Proofs, Proof for All*, David Delahaye, Bruno Woltzenlogel Paleo (eds.), College Publications, Mathematical Logic and Foundations, pp. 171-214.
2. Duží, M., Jespersen B., Materna P. (2010): *Procedural Semantics for Hyperintensional Logic; Foundations and Applications of Transparent Intensional Logic*. Berlin, Heidelberg: Springer.
3. Duží, M., Jespersen, B. (2015): Transparent Quantification into Hyperintensional objectual attitudes. Synthese, vol. 192, No. 3, pp. 635-677.
4. Duží, M. (2012): Extensional logic of hyperintensions. *Lecture Notes in Computer Science*, vol. 7260, pp. 268-290.
5. Duží, M., Materna P. (2012): *TIL jako procedurální logika.* Aleph Bratislava.
6. Gordon, M. J. C., Melhan T. F. (eds).: *Introduction to HOL: A theorem proving environment for higher order logic.* Cambridge University Press, 1993.
7. Moggi, E. (1988): *The Partial Lambda-Calculus*, PhD thesis, University of Edinburg, available as LFCS report at http://www.lfcs.inf.ed.ac.uk/reports/88/ECS-LFCS-88-63/.
8. Tichý, P. (1982): Foundations of partial type theory. *Reports on Mathematical Logic*, vol. 14, pp. 52-72. Reprinted in (Tichý 2004: 467-480).
9. Tichý, P. (1988):*The Foundations of Frege's Logic*. Berlin, New York: De Gruyter.
10. Tichý, P. (2004): *Collected Papers in Logic and Philosophy*, eds. V. Svoboda, B. Jespersen, C. Cheyne. Prague: Filosofia, Czech Academy of Sciences, and Dunedin: University of Otago Press.

Czech Grammar Agreement Dataset for Evaluation of Language Models

Vít Baisa

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

Abstract. AGREE is a dataset and task for evaluation of language models based on grammar agreement in Czech. The dataset consists of sentences with marked suffixes of past tense verbs. The task is to choose the right verb suffix which depends on gender, number and animacy of subject. It is challenging for language models because 1) Czech is morphologically rich, 2) it has relatively free word order, 3) high out-of-vocabulary (OOV) ratio, 4) predicate and subject can be far from each other, 5) subjects can be unexpressed and 6) various semantic rules may apply. The task provides a straightforward and easily reproducible way of evaluating language models on a morphologically rich language.

Key words: language model, grammar agreement, verb suffix, Czech, subject, predicate, dataset, evaluation, perplexity

1 Introduction

Language modeling is one of the most important research fields within natural language processing. Language models are used in many applications ranging from basic tasks as spell checking, diacritic restoration to complex tasks as automatic speech recognition and machine translation.

The vast majority of the improvements have been demonstrated on English since the mainstream datasets are: 1) Brown corpus [1], 2) Penn Treebank [2], 3) Wall Street Journal, 4) English Gigaword [3] and 5) One billion word benchmark (OBB) [4]. All of based on English data.

It can be shown that English is especially suitable for various techniques as it has quite poor morphology and low OOV rate resulting in the fact that relatively small vocabularies are sufficient to cover the vast majority of unseen data.

AGREE task is intended to provide an extrinsic way of evaluating language models on non-English language data. Despite the fact that the Czech subject-predicate agreement is exhibited by verb suffixes, the task is suitable to evaluate both word-based and character-based language models.

The task was inspired by Microsoft Research Sentence Completion Challenge (MSCC) task [5] which contains 1,040 sentences from five Sherlock

Holmes novels by Sir A. C. Doyle and in each sentence, one word is missing and 5 alternatives are supplied. The task is deliberately hard for n-gram models as the missing words are sometimes impossible to guess from a local context.

2 AGREE task

In past tense verbs, subject-predicate agreement is exhibited by grammar suffixes *a*, *o*, *i*, *y* and an empty suffix. These correspond to the following subject types: 1) **-a**, e.g. *žila*: subject is feminine singular (she lived) or neuter plural (kittens lived), 2) **-o**, e.g. *žilo*: neuter singular (a pig lived), 3) **-i**, e.g. *žili*: masculine plural (men lived) or subject is a group of entities of feminine and masculine genders (men and women lived), 4) **-y**, e.g. *žily*: feminine plural (women lived), neuter plural (children lived), 5) **-∅**, žil: masculine singular (he lived).

Other rules mapy apply, e.g. a masculine animate subject outweighs inanimate subjects *muži a stroje pracovali* (men and machines worked) etc.

The nature of the task is similar to MSCC: to choose a suffix properly, the whole sentence must be comprehended. Even though the agreement is grammar-motivated, it depends on the semantics. Sometimes, even the whole sentence might be insufficient to choose a proper suffix.

Unlike in MSCC task, sentences might contain more than one position where a suffix (word) is to be chosen. Commented example sentences with marked verbs follow.

*Pestrý program byl*** vítanou inspirací pro naše soubory.*

In this sentence, the subject *program* (programme) is governed by the predicate *byl* (was). To choose the right word form, language models need to capture just two neighbouring words—bigram.

*Na zpáteční cestě do USA měl*** konvoj vézt zlato, platbu za dodané zbrojní zakázky.*

In this sentence it is enough to look at the subsequent word *konvoj* (convoy). Since Czech has rich morphology, it has free word order so the relative positions of predicates and subjects may vary.

*Určitě tady všichni nešťastnému dědulovi drželi*** palce, ale to bylo*** asi všechno, co pro něho mohli*** udělat.*

Here the first verb is governed by word *všichni* (all of us), which is not neighbouring the verb in past tense. The second is related to the previous word (pronoun *to* (it) is a sign of an anaphora, the anaphoric subject here is the act of keeping one's fingers crossed). The gender of the pronoun is neuter and it is trivial to choose the right word form as the subject and predicate are next to each other. The third is again governed by the main subject (*všichni*). So the agreement is defined by a dependency spanning 12 words.

*Léon Bourgeois navrhl*** i praktický program solidarity.*

In this sentence, the gender of *Léon Bourgeois* needs to be guessed to assign a proper suffix. In the case of masculine, the suffix is empty and in the case of feminine it would be *-a*, i.e. *navrhla*. By omitting low frequency words (*Bourgeois*) which is a standard strategy in word-based techniques, models would probably miss this token and would be unable to learn what verbs with what endings co-occur with it.

*Teď už se normálně postavil*** a ťapkal*** trávou směrem ke mně.*

Some sentences are hard to complete without a proper context. The good strategy here is to assign the same suffix to all verbs since they are usually governed by the same (though sometimes unexpressed) subject. Nevertheless, the sentence indicates that the subject will be something like a cub, a puppy or a kitten. In Czech, puppies and cubs have usually neuter gender. But it might also be a named pet and in this case its real sex is determining the suffix.

*Ve třetím kole narazily*** na celkově třetí Třešňákovou s Pilátovou a podlehly*** jim 0 : 2 (-18, -8).*

Sometimes, common sense might help to guess the right word form. Here it is more probable that the gender of the unexpressed subject is feminine, as the sentence talks about a player couple facing some female opponents (we know it because of the surname endings *-ovou*) in a sport match. It is probable that the players will have the same gender as their opponents as it is usual in sports.

3 AGREE dataset

The dataset consists of 10 million Czech sentences from a Czech Web Corpus [6] with marked verbs in past tense split into three parts: 1) TRAIN with 9,900,000 sentences, 2) VALID with 99,000 sentences and 3) TEST with 996 sentences.

4 Evaluation of various models and a baseline

For the evaluation purpose, all possible combinations are generated yielding 17,940 sentences. TEST set has been manually annotated by several undergraduate students to estimate the difficulty of the task. The average verb accuracy was 86.5.

The baseline model chooses the most frequent word form for each marked verb (the frequencies were extracted from a Czech web corpus). Table 1 contains the summary of accuracies of baseline, random and various word- and character-based models.

The best result (59.8%) has been achieved by a word-based RNN model.[1] RNN performs only slightly better than SRILM 4-gram word-based model [7];

[1] `https://github.com/yandex/faster-rnnlm`

Table 1: The summary of AGREE task results for various models.

Model	Accuracy
Human (average)	86.5
Recurrent Neural Network with hidden layer size 100	59.8
SRILM word-based 4-gram	59.6
Chunk-based language model	58.7
SRILM character-based 9-gram	53.9
Baseline (the most frequent wordform)	42.0
Random (average of 10 runs)	19.6

both with standard settings. The vocabulary was not trimmed: the OOV of testing data against training data caused that in 1,401 sentences (out of 17,940) there was at least one OOV word. This might cause the rather poor results for word-based models .

The character n-gram model has been trained with SRILM, the chunk-based language model operating on byte level is described in [8].

5 Conclusion

AGREE task was developed to promote evaluation of language models focusing on morphologically rich languages. The task is motivated by a morphological phenomenon in Czech language of subject-predicate grammar agreement.

The task is hard as sentences must be comprehended and sometimes even common sense is needed for assigning the suffixes correctly.

Random choice yields 20%, baseline around 40%, best language models achieve 60% and human annotators 90%.

The dataset and auxiliary scripts have been released under Creative Commons Share-alike Attribution licence.[2]

In future we would like to build a similar task for even more morphologically rich languages where the OOV is more pronounced, e.g. Estonian, Hungarian and Turkish.

Acknowledgments. This work has been partly supported by the Masaryk University within the project *Čeština v jednotě synchronie a diachronie – 2016* (MUNI/A/0863/2015). The research leading to these results has received funding from the Norwegian Financial Mechanism 2009–2014 and the Ministry of Education, Youth and Sports under Project Contract no. MSMT-28477/2014 within the HaBiT Project 7F14047.

References

1. Francis, W.N., Kucera, H.: Brown corpus manual. Brown University (1979)

[2] `https://nlp.fi.muni.cz/~xbaisa/agree/`

2. Marcus, M., Marcinkiewicz, M., Santorini, B.: Building a large annotated corpus of English: The Penn Treebank. Computational linguistics **19**(2) (1993) 313–330
3. Graff, D., Kong, J., Chen, K., Maeda, K.: English Gigaword. Linguistic Data Consortium, Philadelphia (2003)
4. Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., Robinson, T.: One billion word benchmark for measuring progress in statistical language modeling. arXiv preprint arXiv:1312.3005 (2013)
5. Zweig, G., Burges, C.J.: The Microsoft Research sentence completion challenge. Technical report, Technical Report MSR-TR-2011-129, Microsoft (2011)
6. Jakubíček, M., Kilgarriff, A., Kovář, V., Rychlý, P., Suchomel, V.: The tenten corpus family. In: 7th International Corpus Linguistics Conference. (2013) 125–127
7. Stolcke, A.: SRILM-an extensible language modeling toolkit. In: Proceedings of the international conference on spoken language processing. Volume 2., Citeseer (2002) 901–904
8. Baisa, V.: Byte level language models. PhD thesis, Masaryk University (11 2016)

Bilingual Logical Analysis of Natural Language Sentences

Marek Medveď, Aleš Horák, and Vojtěch Kovář

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

Abstract. One of the main aims of logical analysis of natural language expressions lies in the task to capture the meaning structures independently on the selected "mean of transport," i.e. on a particular natural language used. Logical analysis should just offer a "bridge between language expressions."
In this paper, we show the preliminary results of automated bilingual logical analysis, namely the analysis of English and Czech sentences. The underlying logical formalism, the Transparent Intensional Logic (TIL), is a representative of a higher-order temporal logic designed to express full meaning relations of natural language expressions.
We present the details of the current development and preparations of the supportive lexicons for the AST (automated semantic analysis) tool when working with a new language, i.e. English. The AST provides an implementation of the Normal Translation Algorithm for TIL aiming to offer a normative logical analysis of the input sentences. We show the similarities and the differences of the resulting logical constructions obtained via both the input languages with a view towards wide bilingual logical analysis.

Key words: semantics, semantic analysis, logical analysis, Transparent Intensional Logic, TIL

1 Introduction

Meaning representations are usually understood as a "dense and shared" form of input sentences. In practical systems, many machine translation tools search for a kind of *interlingua*, a semantic representation used to convey the message content from source to the target language [1,2]. From the theoretical point of view, the meaning of natural language sentences can be fully expressed by using a kind of intensional logic formalism, which allows to represent the clausal relations inherently connected with the referents of real-world concepts. In the current paper, we lean on the formalism of the Transparent Intensional Logic, or TIL [3,4], that allows for a clear logical interpretation of many complex language phenomena.

In the following text, we present the latest results in the design and implementation of automated language-independent semantic analysis of natural language sentences. We show, how the AST tool [5] is adapted for the work with a new syntactic analyser and a new language of the input text. Primarily, AST worked in connection with the Czech language parser `synt` [6], which uses a strict meta-grammar of Czech to strictly decide the correctness of the input. This inevitably leads to lower coverage of borderline phenomena. The modular design of AST is proved via extending the input to processing another parser, namely the SET parser [7], that is based on a flexible pattern-matching dependency concept allowing to process all kinds of input sentences. The SET parser is extendible to other languages, which is demonstrated in this text by comparing the resulting logical constructions of the application of AST and SET to both Czech and English sentences.

In the following sections, we first briefly describe the `synt` and SET parsers with most emphasis on the differences between the parsers, then we enlist the requirements of the AST tool for obtaining the language-dependent lexical information, and finally present the first results of bilingual, i.e. Czech and English, logical analysis within the AST tool.

2 Parsing

The semantic processing of natural language sentence builds upon the result of structural syntactic analysis or *parsing*. As a prevalent type of presenting the hierarchical organization of the input sentence most parsers are able to provide a comprehensive representation in a form of a syntactic tree, which is also the form processed by the AST tool.

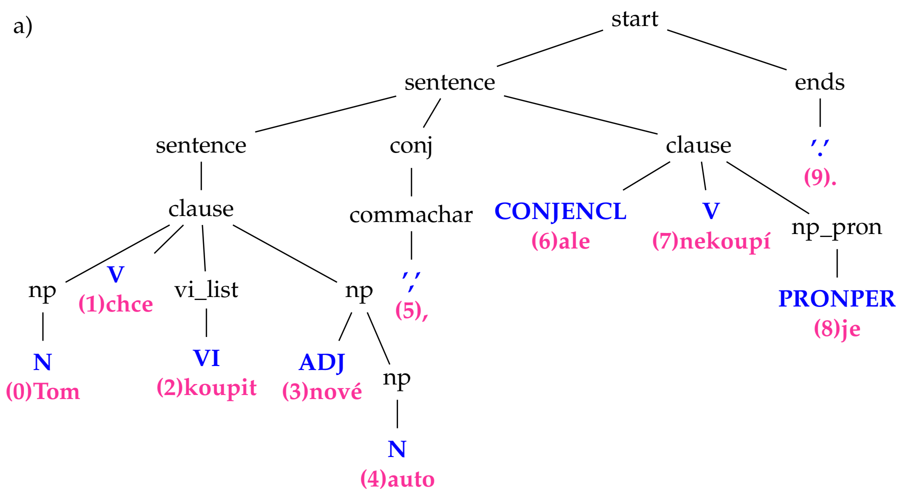
Both `synt` and SET parsers are rule-based systems but they use different approach in providing syntactic analysis, which also influences the differences of their syntactic trees.

2.1 The `synt` parser

The `synt` parser [6,8] was developed around the meta-grammar concept based on traditional linguistic rule systems. The core of the parser uses a context-free grammar with contextual actions and performs a stochastic agenda-based head-driven chart analysis.

The internal representation concentrates on fast processing of very ambiguous syntactic structures. The parser is able to process sentences with syntactic combinations resulting in millions of possible syntactic trees with an average processing time of 0.07 s per sentence. The analysis trees are stored in a *packed shared forest*, which can be transformed to several kinds of output such as an ordered list of syntactic trees, a dependency graph or an annotated list of extracted phrases.

The `synt` parser is also able to perform a TIL-based logical analysis of a sentence as a first implementation of the Normal Translation Algorithm a)



b) $$\lambda w_1 \lambda t_2 \Big((\exists x_3)(\exists i_4)(\exists i_5) \Big([\textbf{Does}_{w_1 t_2}, i_5, [\textbf{Imp}_{w_1}, x_3]] \wedge [[\textbf{nový}, \textbf{auto}]_{w_1 t_2}, i_4] \wedge x_3 = [\textbf{chtít}, i_4]_{w_1} \wedge [\textbf{Tom}_{w_1 t_2}, i_5] \Big) \wedge \Big[\textbf{Not}, \Big[\textbf{True}_{w_1 t_2}, \lambda w_6 \lambda t_7 (\exists x_8)(\exists i_9) \Big([\textbf{Does}_{w_6 t_7}, On, [\textbf{Perf}_{w_6}, x_8]] \wedge [\textbf{on}_{w_6 t_7}, i_9] \wedge x_8 = [\textbf{koupit}, i_9]_{w_6} \Big) \Big] \Big] \Big) \ldots \pi$$

Fig. 1: The synt outputs for the sentence: "Tom chce koupit nové auto, ale nekoupí je." (Tom wants to buy a new car, but he will not buy it.), a) the syntactic tree, b) the logical construction.

(NTA [9]), see an example of both the synt tree output and the logical construction in Figure 1. The logical analysis in synt is tightly connected to the parsing process and is not directly transferable to other representations, which was the reason for design and implementation of the parser-independent AST tool.

2.2 SET parser

The SET parser[1] [7] is a rule-based parser based on pattern-matching dependency rules. The SET grammar consists of a set of pattern matching specifications that compete with each other in the process of analysis. From the best matches, SET builds a full coverage syntactic dependency tree of the input sentence. Currently, the system includes grammars for Czech, Slovak and English, each with few dozens of rules that sufficiently model the syntax of the particular language, thanks to the expressive character of the formalism.

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

```
TMPL: $ATTR $...* noun  AGREE 0 2 cgn   MARK 0  DEP 2   PROB 6000
    LABEL modifier
    LABEL rule_sch ( $$ $@ "[#1,#2]" )
```

Fig. 2: SET rule for adjective-noun modifier (e.g. "black dog") supplemented with a TIL schema. The `$ATTR` and `noun` aliases are defined in other parts of the grammar. Actions require agreement in case, gender and number (`cgn`), create an *adjective* $\rightarrow$ *noun* dependency and set up heavy weight (`PROB`) for this rule. The TIL schema says that the attribute (#1) is applied as a function to the noun (#2) – these indexes refer directly to the resulting structured tree.

Due to the nature of the parsing process (each word is in the end included into the tree structure), the parser has 100% coverage, i.e. it provides an analysis for any input sentence.

The input format for the parser is a morphologically annotated sentence in the vertical format.[2] The output options include hybrid trees,[3] dependency trees, phrase structure trees, or a "bush" output.[4]

2.3 SET modifications for TIL analysis

The phrase structure output of SET was the most suitable for the TIL analysis, because the current AST system also works with a phrase structure tree. However, as opposed to the `synt` system, there is no formal grammar (in the Chomsky's sense) according to which the phrase structure tree was created; it is just a conversion from the hybrid tree (that was created according to the SET pattern-matching grammar). Especially, it is not clear what types of phrasal sub-trees can appear in the result, e.g. what components can a noun-phrase non-terminal have; in theory, the number of options is unlimited.

This was a problem, as the TIL analysis algorithm processes the tree level by level and it is to these levels (that correspond to grammar rules in the `synt` parser) where the particular schemata for TIL are assigned. Therefore, we have developed a new type of SET phrase structure output[5] where each level of the tree corresponds exactly to one rule in the SET grammar. This adaptation allows to assign the TIL logical construction schemata to be assigned to the rules in the SET grammar.

With this concept in mind, the Czech and English SET grammars[6] were supplemented with the schemata for TIL analysis exported to the new phrase

[2] Plain text tabular representation of a tokenized and annotated text.

[3] Syntactic tree format that combines dependency and phrase structure features; this is the primary output of the parser and all the other outputs are based on this tree

[4] A "bush" denotes a structure of phrases connected with dependencies, see [10] for details.

[5] as a new option `-s` which stands for "structured"

[6] denoted as `tilcz.set` and `tilen.set` in the repository

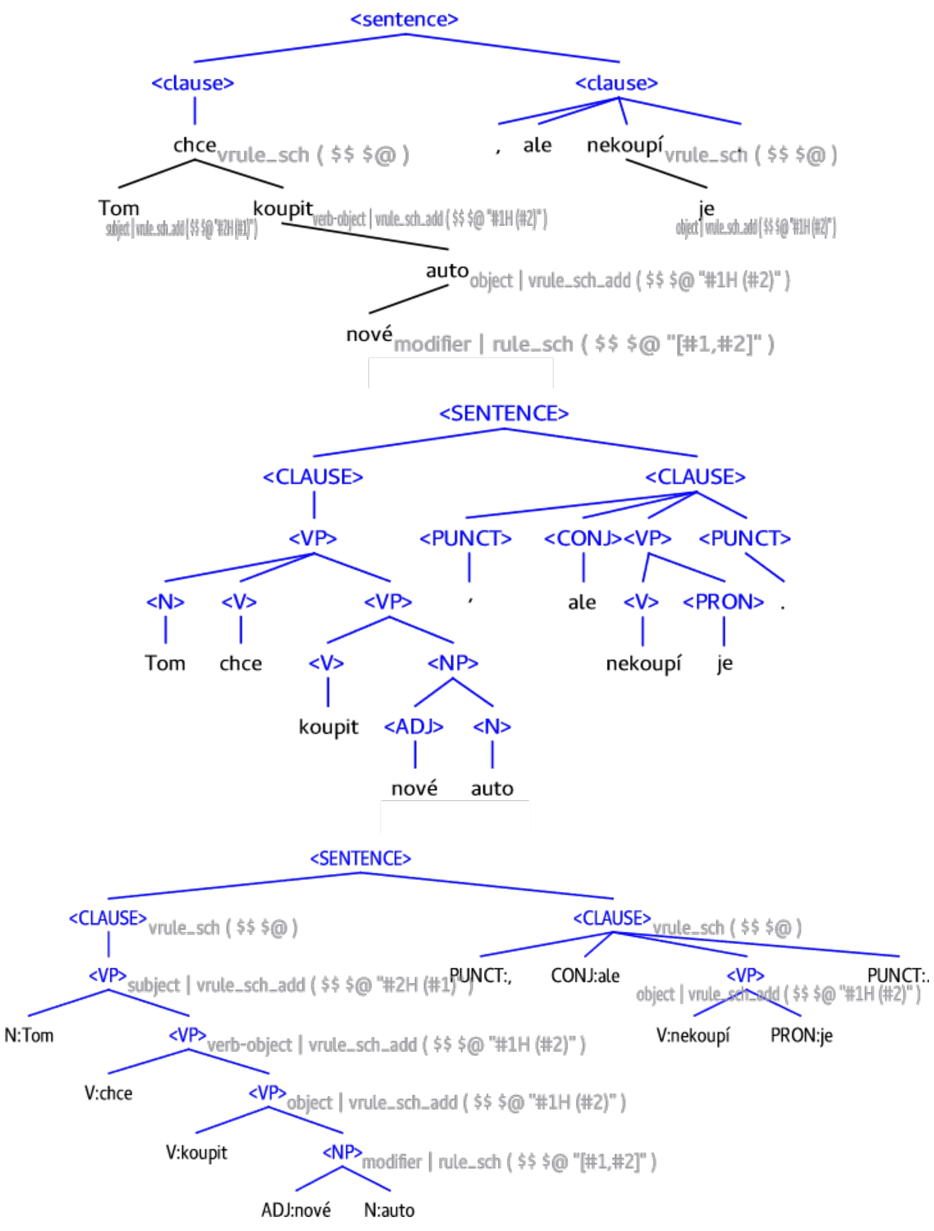

<sentence>
<clause>
<clause>
chce vrule_sch ($$ $@)
, ale nekoupí vrule_sch ($$ $@)
Tom subject | vrule_sch_add ($$ $@ "#2H (#1)")
koupit verb-object | vrule_sch_add ($$ $@ "#1H (#2)")
je object | vrule_sch_add ($$ $@ "#1H (#2)")
auto object | vrule_sch_add ($$ $@ "#1H (#2)")
nové modifier | rule_sch ($$ $@ "[#1,#2]")
<SENTENCE>
<CLAUSE>
<CLAUSE>
<VP>
<PUNCT>
<CONJ><VP>
<PUNCT>
<N>
<V>
<VP>
,
ale
<V>
<PRON>
.
Tom
chce
<V>
<NP>
nekoupí
je
koupit
<ADJ>
<N>
nové
auto
<SENTENCE>
<CLAUSE> vrule_sch ($$ $@)
<CLAUSE> vrule_sch ($$ $@)
<VP> subject | vrule_sch_add ($$ $@ "#2H (#1)")
PUNCT:,
CONJ:ale
<VP> object | vrule_sch_add ($$ $@ "#1H (#2)")
PUNCT:.
N:Tom
<VP> verb-object | vrule_sch_add ($$ $@ "#1H (#2)")
V:nekoupí
PRON:je
V:chce
<VP> object | vrule_sch_add ($$ $@ "#1H (#2)")
V:koupit
<NP> modifier | rule_sch ($$ $@ "[#1,#2]")
ADJ:nové
N:auto

Fig. 3: Different SET trees for the sentence "Tom chce koupit nové auto, ale nekoupí je." (Tom wants to buy a new car but he will not buy it.) The hybrid tree (1) has TIL schemata assigned to particular nodes (according to the SET grammar), in the default phrasal tree (2), it was not easy to decide where the schemata should be stored, so we introduced a "structured" tree (3) where the schemata are assigned to non-terminals (with the same semantics as in (1)).

structure output of the SET syntactic tree. Figure 2 shows an example of a SET rule annotated with a TIL schema exploited by the AST system. In Figure 3, we can see the difference between the default SET hybrid tree, the default phrase structure tree and the new "structured" tree. TIL schemata assigned to the particular rules are shown in grey within these trees.

As follows from Figures 3 and 1, the structure of the `synt` and SET trees is different. For example, the clause level in `synt` trees contains the main verb and the modal verb on the same level; the SET tree uses binary branching. Therefore, the AST tool needed to be customized to work with the SET "structured" trees.

3 The AST Modules

AST is a language independent tool for semantic analysis of natural language sentences. The ideas of the AST implementation emerged from the `synt` parser while targeting at easy adaptations for new languages and syntactic analyser. AST follows the principles of the NTA algorithm and uses the Transparent Intentional Logic (TIL) for the formalization of semantic constructions.

The AST tool is a modular system and consists of seven main parts, which mostly correspond to the required language-dependent lexicons:[7]

– the *input processor*: reads the input with a syntactic tree in textual form (see an example of SET tree in Figure 4).
– the *grammar processor*: reads the list of grammar rules and schema actions to obtain a schema for each node inside the tree (only for `synt` parser). An example of such rule is:

```
np -> left_modif np
    rule_schema ( "[#1,#2]" )
```

  In this case the resulting logical construction of the *left-hand side* `np` is obtained as a (logical) application of the `left_modif` (sub)construction to the *right-hand side* `np` (sub)construction.
– the *lexical items processor*: a list of lexical item specifications with TIL types for each leaf (word) in the tree structure. A lexical item example for the verb "koupit" (buy) is:

  `koupit`
  `/k5/otriv` $(((o(oo_{\tau\omega})(oo_{\tau\omega}))\omega)\iota)$

– the *schema processor*: general module for operations over the logical construction schemata. An example of creating the construction for the noun phrase "nové auto" (new car):

  `rule_schema: '[#1,#2]'`
  `Processing schema with params:`
  `#1:` ^{0}`nový...`$((o\iota)_{\tau\omega}(o\iota)_{\tau\omega})$
  `#2:` ^{0}`auto...`$(o\iota)_{\tau\omega}$
  `Resulting constructions:`
  `[`^{0}`nový/`$((o\iota)_{\tau\omega}(o\iota)_{\tau\omega})$`,` ^{0}`auto/`$(o\iota)_{\tau\omega}$`]...`$(o\iota)_{\tau\omega}$

[7] See [5] for details.

- the *verb valency processor*: identifies the correct valency frame for the given sentence and triggers the schema parser on sub-constructions according to the schema coming with the valency. An example for the verb "koupit" (eat) is as follows

  ```
  koupit
  hPTc4 :exists:V(v):V(v):and:V(v)=[[#0,try(#1)],V(w)]
  ```

- the *prepositional valency expression processor*: translates analysed prepositional phrases to possible valency expressions. For instance, the record for the preposition "k" (to) is displayed as

  ```
  k
  3 hA hH
  ```

  The preposition "k" (to) can introduce a prepositional phrase of a *where-to* direction `hA`, or a modal *how/what* specification `hH`.
- the *sentence schema processor*: if the sentence structure contains subordination or coordination clauses the sentence schema parser is triggered. The sentence schemata are classified by the conjunctions used between clauses. An example for the conjunction "ale" (but) is:

  ```
  ("";"ale") : "lwt(awt(#1) and awt(#2))"
  ```

  The resulting construction builds a logical conjunction of the two clauses as can be seen in the Figure 1 example.

3.1 AST Adaptations for the SET Trees

As explained above in Section 2.3, the dependency core of the SET parser defies to the logical schema prescription used in the case of the `synt` parser. The final solution implemented in AST is based on two ideas:

- organization of the SET output in an adapted phrasal-dependency tree, viz the "structured" tree in Figure 3,
- splitting the corresponding *verb rule schemata* to binary additive actions denoted as `vrule_sch_add`.

An example of the resulting labeled tree form is displayed in Figure 4, where each constructive edge obtains a logical schema label from the SET grammar.

In the `synt` tree, all constituent groups are reachable directly from the clause node (see Figure 1). In the SET tree, the additive actions `vrule_sch_add` build the arguments needed by the clause-level `vrule_sch` (*verb rule schema*) action. The action arguments denote different constituent types: a main verb, an auxiliary constituent or a clause constituent. The schemata also distinguish whether the reference denotes the whole (sub)phrase or applies to the headword only (tag `H`). By this process, the identified constituents are unified with the list of

```
id word:nterm lemma  tag          pid til schema
0  N:Tom       Tom    k1gMnSc1;ca14 p
1  V:chce      chtít  k5eAaImIp3nS  15
2  V:koupit    koupit k5eAaPmF      16
3  ADJ:nové    nový   k2eAgNnSc4d1  17
4  N:auto      auto   k1gNnSc4      17
5  PUNCT:,     ,      kIx           10
6  CONJ:ale    ale    k8xC          10
7  V:nekoupí   koupit k5eNaPmIp3nS  13
8  PRON:je     on     k3xPp3gNnSc4  13
9  PUNCT:.     .      kIx.          10
10 <CLAUSE>           k5eNaPmIp3nS  12  vrule_sch ( $$ $@ )
11 <CLAUSE>           k5eAaImIp3nS  12  vrule_sch ( $$ $@ )
12 <SENTENCE>                       -1
13 <VP>        koupit k5eNaPmIp3nS  10  vrule_sch_add ( $$ $@ "#1H (#2)" )
14 <VP>        chtít  k5eAaImIp3nS  11  vrule_sch_add ( $$ $@ "#2H (#1)" )
15 <VP>        chtít  k5eAaImIp3nS  14  vrule_sch_add ( $$ $@ "#1H (#2)" )
16 <VP>        koupit k5eAaPmF      15  vrule_sch_add ( $$ $@ "#1H (#2)" )
17 <NP>        auto   k1gNnSc4      16  rule_sch ( $$ $@ "[#1,#2]" )
```

Fig. 4: SET tree format

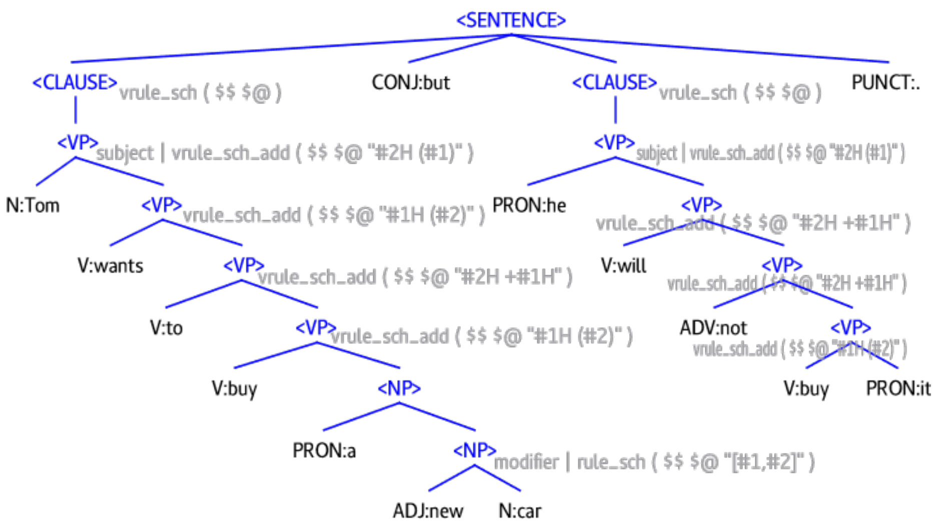
possible verb valency frames, which finally leads to the selected clause logical schema.

On the sentence level, where the TIL constructions of all clauses are combined into the final construction, the synt grammar specified action for *sentence rule schema*, which consults the sentence schema lexicon to determine the construction schema. In case of SET, there is no top-level rule, that is why AST needs to automatically trigger the function for a recursive combination of clauses in the SET tree output.

4 Bilingual Logical Analysis

As we have mentioned above, the SET parser contains comparable grammars for the Czech and English languages which in combination with the above described modifications of both SET and AST systems enables bilingual logical analysis. The complexity of the two grammars is very similar – the Czech grammar currently contains 71 rules, the English one has 45 rules. Both the grammars were annotated with TIL schemata consistently.

The English language processing in AST is, however, still preliminary in the sense that the required logical lexicons are filled with testing data only. The example logical constructions obtained from translated sentences show promising regularities in the construction structures, which allow for isomorphic concept labeling between the two languages. An example of the



<SENTENCE>
<CLAUSE> vrule_sch ($$ $@)
CONJ:but
<CLAUSE> vrule_sch ($$ $@)
PUNCT:.
<VP> subject | vrule_sch_add ($$ $@ "#2H (#1)")
<VP> subject | vrule_sch_add ($$ $@ "#2H (#1)")
N:Tom
<VP> vrule_sch_add ($$ $@ "#1H (#2)")
PRON:he
<VP> vrule_sch_add ($$ $@ "#2H +#1H")
V:wants
<VP> vrule_sch_add ($$ $@ "#2H +#1H")
V:will
<VP> vrule_sch_add ($$ $@ "#2H +#1H")
V:to
<VP> vrule_sch_add ($$ $@ "#1H (#2)")
ADV:not
<VP> vrule_sch_add ($$ $@ "#1H (#2)")
V:buy
<NP>
V:buy
PRON:it
PRON:a
<NP> modifier | rule_sch ($$ $@ "[#1,#2]")
ADJ:new
N:car

Fig. 5: SET tree for English sentence "Tom wants to buy a new car but he will not buy it.", according to SET grammar for English, and annotated with TIL schemata.

SET tree for the English variant of the original Czech sentence is displayed in Figure 5, which leads to the logical construction of

$$\lambda w_1 \lambda t_2 \Big((\exists x_3)(\exists i_4)(\exists i_5) \Big([\mathbf{Does}_{w_1 t_2}, i_5, [\mathbf{Imp}_{w_1}, x_3]] \wedge [[\mathbf{new}, \mathbf{car}]_{w_1 t_2}, i_4] \\ \wedge\ x_3 = [\mathbf{to_want}, i_4]_{w_1} \wedge [\mathbf{Tom}_{w_1 t_2}, i_5] \Big) \wedge \Big[\mathbf{Not}, \Big[\mathbf{True}_{w_1 t_2}, \\ \lambda w_6 \lambda t_7 (\exists x_8)(\exists i_9) \Big([\mathbf{Does}_{w_6 t_7}, He, [\mathbf{Perf}_{w_6}, x_8]] \wedge [\mathbf{it}_{w_6 t_7}, i_9] \\ \wedge\ x_8 = [\mathbf{to_buy}, i_9]_{w_6} \Big) \Big] \Big] \Big) \ldots \pi$$

The resulting construction can be directly compared with the TIL logical construction from Figure 1b).

5 Conclusions and Future Directions

Representing the essence of the structural meaning via an automated process offers a valuable tool for semantic processing of natural language texts. The validity of such representation can be verified with the level of correspondence in the resulting logical formulae when processing direct translations between two natural languages.

In this paper, we have presented the current development of the language-independent automated semantic tool AST, which shows the capabilities of logical analysis for two languages – Czech and English. Even though the

English analysis currently does not cover standard vocabulary on the same level as the Czech analysis, the preliminary results are promising in the structural correspondences between the logical representations.

In the future research, the required English TIL lexicons will be connected to large available resources such as VerbNet, FrameNet and WordNet, which will allow to obtain the same level of coverage for both languages. The logical correspondences will be then thoroughly evaluated on a representative set of bilingual sentence pairs.

Acknowledgments. This work has been partly supported by the Czech Science Foundation under the project GA15-13277S.

References

1. Anismovich, K., Selegey, V., Zuev, K.: Systems for translating sentences between languages using language-independent semantic structures and ratings of syntactic constructions (July 3 2012) US Patent 8,214,199.
2. Popel, M., Žabokrtský, Z.: TectoMT: Modular NLP Framework. In: International Conference on Natural Language Processing, Springer (2010) 293–304
3. Tichý, P.: The foundations of Frege's logic. Walter de Gruyter, New York (1988)
4. Duží, M., Jespersen, B., Materna, P.: Procedural Semantics for Hyperintensional Logic. Foundations and Applications of Transparent Intensional Logic. Volume 17 of Logic, Epistemology and the Unity of Science. Springer, Berlin (2010)
5. Medveď, M., Horák, A., et al.: AST: New Tool for Logical Analysis of Sentences based on Transparent Intensional Logic. (2015)
6. Horák, A.: Computer Processing of Czech Syntax and Semantics. Librix.eu, Brno, Czech Republic (2008)
7. 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. Volume 6562 of Lecture Notes in Computer Science., Berlin, Springer (2011) 161–171
8. Jakubíček, M., Horák, A., Kovář, V.: Mining phrases from syntactic analysis. In: Text, Speech and Dialogue. (2009) 124–130
9. Horák, A.: The Normal Translation Algorithm in Transparent Intensional Logic for Czech. PhD thesis, Faculty of Informatics, Masaryk University, Brno (2002)
10. Grác, M.: Case study of BushBank concept. In: The 25th Pacific Asia Conference on Language, Information and Computation, Singapore, Institute for Digital Enhancement of Cognitive Development, Waseda University (2011) 353–361

ScaleText: The Design of a Scalable, Adaptable and User-Friendly Document System for Similarity Searches

Digging for Nuggets of Wisdom in Text

Jan Rygl[1], Petr Sojka[2], Michal Růžička[2], and Radim Řehůřek[1]

[1] RaRe Technologies, {jimmy,radim}@rare-technologies.com
[2] Faculty of Informatics, Masaryk University, Brno, Czech Republic
sojka@fi.muni.cz, ORCID: 0000-0002-5768-4007
and mruzicka@mail.muni.cz, ORCID: 0000-0001-5547-8720

Abstract. This paper describes the design of a new ScaleText system aimed at scalable semantic indexing of heterogeneous textual corpora. We discuss the design decisions that lead to a modular system architecture for indexing and searching using semantic vectors of document segments – nuggets of wisdom. The prototype system implementation is evaluated by applying Latent Semantic Indexing (LSI) on the Enron corpus. And the Bpref measure is used to automate comparing the performance of different algorithms and system configurations.

Key words: ScaleText, vector space modelling, Latent Semantic Indexing, LSI, machine learning, scalable search, search system design, text mining

1 Introduction

Today's growing information overload dictates the need for effective semantic searching in custom datasets, such as emails, texts in corporation information systems and knowledge bases, Wikipedia, web browsing history, and in personal information space. Such a search service gives working professionals a competitive advantage, and allows them to have relevant information at their fingertips.

Content semantics indexing is the king for document indexing and filtering large volumes of textual data. Its relevance search goes beyond string, word or phrase indexing.

In this paper we describe the design of ScaleText, a system that aims to meet the demands of the working professional's information search needs. The design imperatives are:

Scalability: with the size of today's document collections, efficiency is a primary concern, allowing low latency responses.

Adaptability: since no size fits all, the system should be easily customizable and tunable for any given application purpose.
Relevance: search precision could be improved by clever semantic representations of the meanings of indexed texts. It is both necessary and desirable to find highly relevant document chunks.
Implementation **C**larity: the implementation should be written with ease of maintenance in mind.
Simplicity: keep it simple stupid, yet provide the functionality needed.

Having SARICS in mind during the design phase lays the foundation for a system capable of meeting the *big data* needs of many enterprises.

This paper is structured as follows: Section 2 evaluates state-of-the-art systems and approaches. In Sections 3 and 4 the top-level system architecture is described, with processing pipelines for indexing and searching, the main components of ScaleText. Section 5 describes automatic evaluation framework for human-unassisted comparison of implementations and configurations of our prototype system on the ground truth of Enron corpus. We conclude with an overview of our contributions along with our work plans for the future in Section 6.

2 The State of the Art in Semantic Document Processing

There has been a noticeable drift away from an emphasis on keyword-based statistics such as term frequency–inverse document frequency (TfIdf) weighting to semantic-based methods such as Latent Semantic Indexing LSI [4] or semantic vectors learned by deep learning [7]. Improving the relevance of search results and scalability are also current design imperatives, given that semantic searches are often performed during web browsing [10] and even at each keystroke as is the case with Google Instant.

Approaches to Semantics The key to better relevance is some sort of sense representation of words, phrases, sentences, paragraphs and documents. We can have either (i) *a discrete representation* of meaning, which can be based on knowledge-based representations such as WordNet, BabelNet, Freebase or Wikipedia, or (ii) *a smooth representation* based on a distributional hypothesis, e.g. representing meanings as word, phrase, sentence,... embeddings [7] which are learned from the language used in big corpora by unsupervised, deep learning approaches, or by topic modeling [1].

Existing Frameworks and Systems There are already frameworks that support building smooth semantic models such as Gensim [8]. One system that builds on semantic document models is Kvasir [10] which supports instant searching in the web browser, thereby stressing the need for speed and relevance.

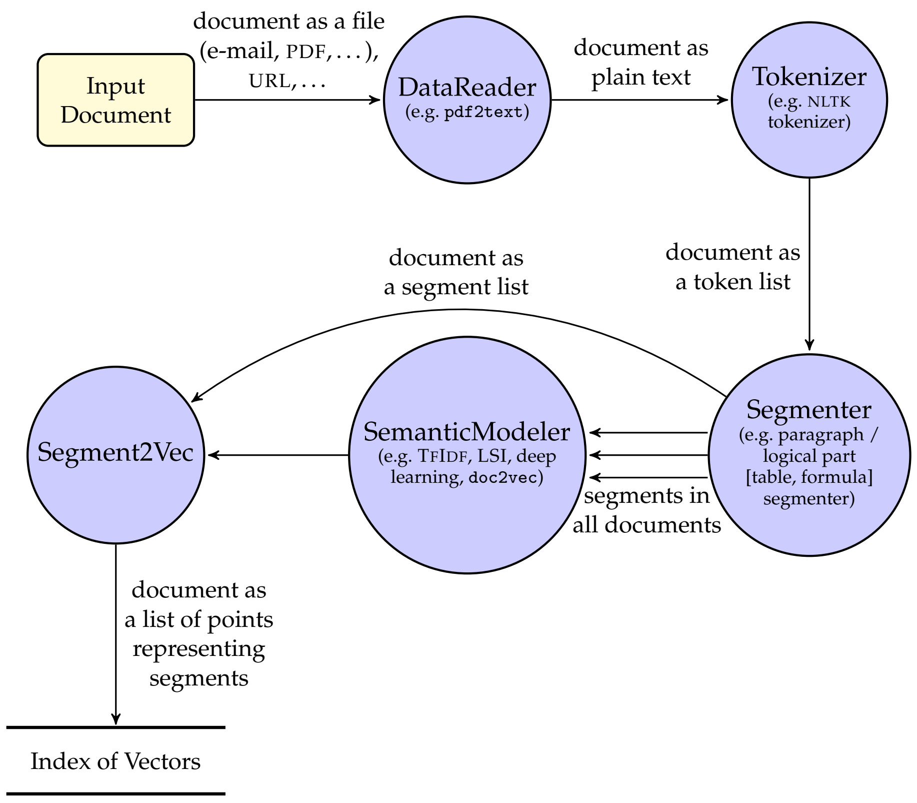

Input Document
document as a file (e-mail, PDF, ...), URL, ...
DataReader
(e.g. pdf2text)
document as plain text
Tokenizer
(e.g. NLTK tokenizer)
document as a token list
Segmenter
(e.g. paragraph / logical part [table, formula] segmenter)
document as a segment list
segments in all documents
SemanticModeler
(e.g. TFIDF, LSI, deep learning, doc2vec)
Segment2Vec
document as a list of points representing segments
Index of Vectors

Fig. 1: Data flow diagram of document indexing in ScaleText

As there are still many open questions to be clarified in text semantics representation, our ScaleText system architecture has been designed to be modular, based on a set of components with a defined purpose and communication interfaces. Different module implementations give the system extra flexibility. Indexing and searching, as main components of ScaleText, are outlined in the following sections.

3 Indexing: Storing Document Chunks as Points in Vector Space

ScaleText introduces a flexible data processing pipeline for document indexing, leading to semantic document representations in a vector space. The overall scheme of document transformations in the indexing workflow is depicted in Figure 1.

In the course of our SARICS imperatives, ScaleText is flexible what format of document is accepted on its input. Raw input documents are read from their primary resources outside ScaleText by the `DataReader` module. Different implementations of the `DataReader` component can be used to provide data from arbitrary data sources such as log files, data files, binary streams, etc. and in various formats such as plain text documents, PDF documents, emails with attachments. Every raw input document is transformed into plain full text, and given a unique document identifier (`doc_id`). All data are made persistent in the `Storage` module.

The `Tokenizer` module processes plain text with a standard linguistic pipeline: (i) tokenization, part of speech tagging, and (ii) phrase detection all take place at this stage. The next stage in the document processing is segmentation in the `Segmenter` module. Plain text is cut into a list of small, meaningful segments, called *nuggets*. Nuggets usually take the form of a triplet consisting of a document identifier (`doc_id`), a segment identifier (`seg_id`) and a list of tokens (`tokens`) comprising a paragraph or equation or another logic element with semantic meaning, extracted from the document plain text. Segmenting a document into nuggets is one of the key ScaleText design ideas that facilitates the semantic indexing of textual data.

Having nuggets of all documents enables us to build a semantic model of an indexed dataset. To represent the semantics of the documents we can use distributional semantics modeling, topic modeling methods, deeply learned representations or LSI. The semantic document model can be rebuilt and retrained at any time from the currently indexed nuggets. In our default implementation, the TfIdf (Term frequency–Inverse document frequency) matrix is computed from the `tokens` of all nuggets, followed by LSI, which results in the projection of nuggets into a latent semantic subspace.

Thus, every document is represented as a list of semantic vectors of nuggets. Both the model and the vectors are persistently stored in a database and used during the search.

4 Document Similarity Search: Digging for Nuggets of Wisdom

The indexed dataset is used for similarity searching. To pursue the gold mining metaphor, gold nuggets are washed with different gold mining techniques. The overall schema of the search procedure is depicted in Figure 2.

Query representation as nuggets The query document is segmented into nuggets.

Querying and scoring We build a set of hit lists between the query and all database nuggets. The hit lists are in the form of triplets of `doc_id` and `seg_id`, referring to a database nugget, and a `score` – a numerical represention of its semantic similarity to the query nugget. Cosine similarity is the implicit similarity measure.

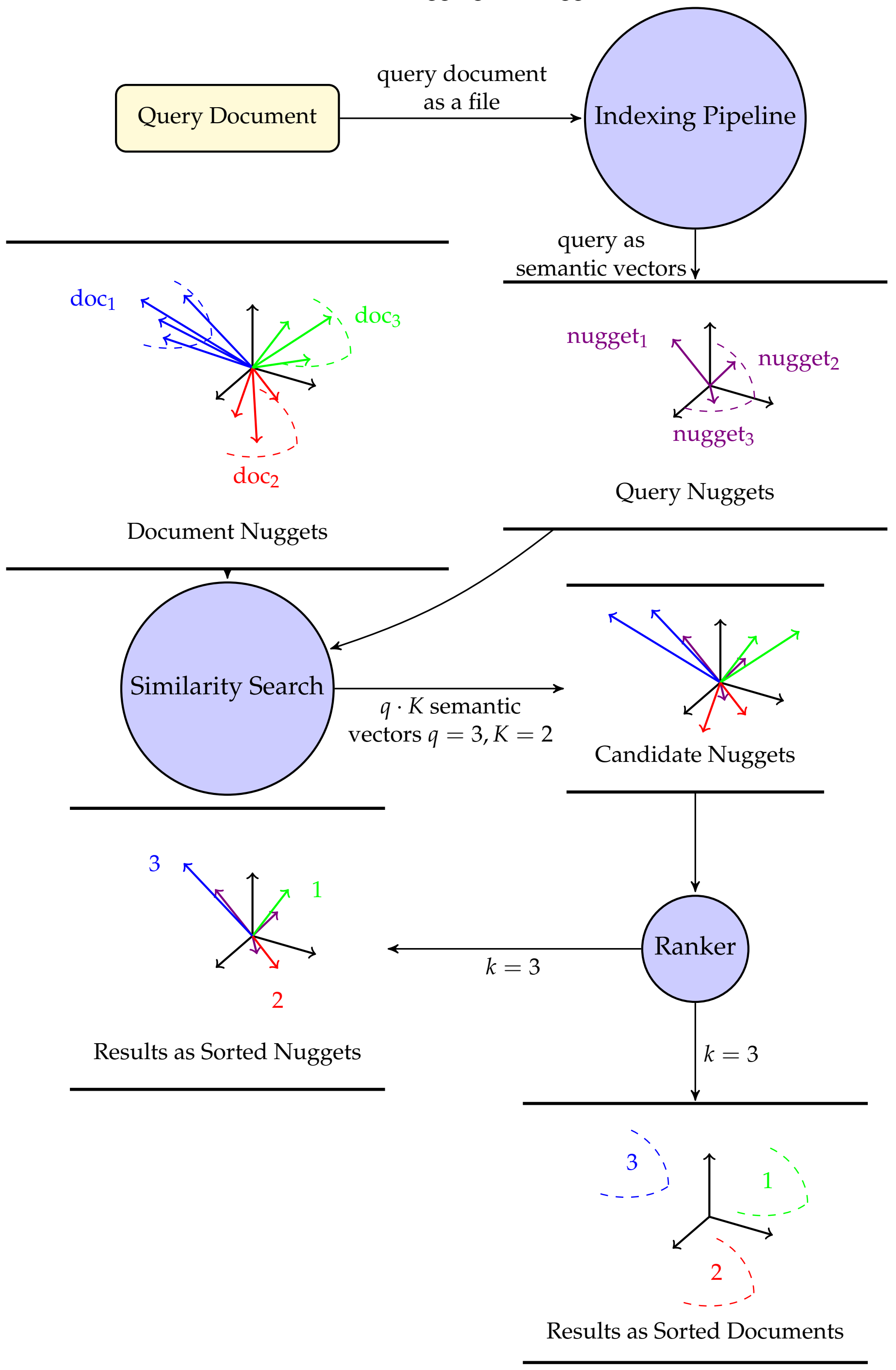


Fig. 2: Data flow diagram of document similarity search in ScaleText. q is the number of query nuggets, K is the number of best nugget candidates for each query nugget, and k is the number of desired final results

Hit merging and sorting strategies The final step of the search procedure is merging and sorting the nuggets found.

The implicit strategy sorts the nuggets by the value of the similarity score only. More advanced strategies can boost scores of the nuggets, for example, based on the number of matching nuggets belonging to the same document. In this case, the score of document nuggets also depends on the overall coverage of the document by the query nuggets.

Document-based result sorting When users prefer whole documents to individual nuggets, the results can be the documents sorted by an aggregation of matched nugget scores per document. There are various aggregation possibilities, such as arithmetic mean, maximum, or sum normalized by the document length.

Scoring and sorting depend on data and application goals but ScaleText provides a flexible architecture to achieve them: a separate `Ranker` module for reordering the results – found nuggets – allows a suitable results sorting strategy to be implemented, tailored to a particular user's needs. The variability of nugget mining strategies that ScaleText design offers provides an opportunity to fine-tune the system with respect to the needs and specifics of a particular project and dataset.

5 Automatic Evaluation Framework for System Modules

As Figures 1 and 2 show, different implementations of the modules in ScaleText data processing workflow can be used. It is necessary to evaluate the system performance of different configurations. In order to achieve a quick verification of ScaleText design ideas and rapid prototyping, we needed a fully automatic, fast and human-unassisted evaluation procedure to compare exchangeable modules in the architecture. We consequently built a ScaleText prototype on top of existing libraries such as Gensim [8] and Spotify Annoy[3], using agile development techniques.

To measure and compare performance of the system modules we needed a dataset with ground truth for a set of queries.

Evaluation dataset We used TREC 2010 Legal track version [3, chapter 2] of the Enron dataset [9]: 455,449 messages plus 230,143 attachments form the 685,592 documents of the TREC 2010 Legal Track collection.

Ground truth dataset To build our ground truth we exploited the availability of 2,720 documents out of the Enron dataset which had an assessed relevance to the Learning task [3, chapter 4.1] that consisted of 8 topics. Every ground truth document was labeled as relevant or irrelevant to each of the topics.

[3] `https://github.com/spotify/annoy`

Query dataset ScaleText uses whole documents as queries for similarity searches. For automatic prototype evaluation we used our ground truth documents, i.e. documents with known relevance to the topics, as the queries.

Bpref@*k* evaluation metric We used the Bpref measure to evaluate the instance performance of the ScaleText system. It is a cheap and rigorous measure of the performance effect of module changes which does not need to assume the completeness of the relevance judgments in the evaluation dataset.

However, the original Bpref proposal [2] was found [6] not to correspond to the actual `trac_eval`[4] implementation of Bpref. Furthermore, the `trac_eval` implementation still does not work correctly on result lists where the number k of inspected results (Bref@k) is lower than the number of relevant results by ground truth. To cope with this we finally implemented Bpref@k as follows:

$$\text{Bpref@}k = \frac{1}{\min(R,k)} \sum_r \left(1 - \frac{\min\left(\text{number of } n \text{ ranked higher than } r, R\right)}{\min(N,R)}\right),$$

where R is the number of documents relevant to the topic, N is the number of documents irrelevant to the topic, k is the maximal number of inspected results, and "number of n ranked higher than r" is the number of irrelevant documents (according to the judgment) ranked higher than the relevant (according to the judgment) document r that is being processed in the step.

Evaluation procedure For a given query, every document in the result list is automatically classified as (i) *relevant*, if the document is in our ground truth and the relevance assessment for the topic is the same as for the query document, (ii) *irrelevant*, if it is in our ground truth but has a different relevance assessment from the query, (iii) *unknown* otherwise.

Table 1 provides examples of the evaluations of different ScaleText configurations and ranking strategies. It is important to note that unsupervised machine learning techniques have been used, i.e. Enron seed set [2] was *not* used to train the model on labeled data. This evaluation procedure is useful for a fast, automatic, human-unassisted evaluation of system configuration effectiveness, as it shows the differences among configurations. E.g., from the Bpref@100 values it is clear that setting the number of LSI features as high as 500 already degrades performance. Also, TfIdf+LSI configuration gives significantly better results than using only TfIdf weighting.

6 Conclusion and Future Work

We have designed a ScaleText system for scalable semantic searching and indexing. The system has been designed with SARICS (Scalability, Adaptability, Relevance, Implementation, Clarity and Simplicity) in mind. Its architecture

[4] `http://trec.nist.gov/trec_eval/`

Table 1: ScaleText prototype evaluation on the Enron dataset via Bpref. The single metric value is the average of Bpref@100 over all the queries

document model	document ranking strategy	#features	avg. Bpref@100
TfIdf	maximum nugget score	100	0.0451
TfIdf+LSI	maximum nugget score	50	0.0460
TfIdf+LSI	maximum nugget score	100	0.0565
TfIdf+LSI	maximum nugget score	500	0.0358
TfIdf	average nugget score	100	0.0451
TfIdf+LSI	average nugget score	50	0.0460
TfIdf+LSI	average nugget score	100	0.0548
TfIdf+LSI	average nugget score	500	0.0358
TfIdf	normalized sum of nugget scores	100	0.0451
TfIdf+LSI	normalized sum of nugget scores	50	0.0460
TfIdf+LSI	normalized sum of nugget scores	100	0.0534
TfIdf+LSI	normalized sum of nugget scores	500	0.0358

allows for massive parallelization of both crucial operations – indexing and searching with low latency, yet allowing easy maintenance and pluggable modules for semantic indexing (LSI, distributive semantics) and searching (*k*-NN search, new techniques for searching in feature vector spaces).

We have implemented ScaleText in Python for easy prototyping and maintenance. Semantic modeling algorithms use a high-performance implementation of Gensim.

We have designed a mechanism for evaluating pluggable system modules. The ground truth Enron database with Bpref metric has allowed us to quickly and automatically measure the performance of the system and compare the module effectiveness of different system configurations.

In subsequent work we will evaluate different vector space representations of documents and prepare a methodology for configuring ScaleText to meet document system search demands of both the research community and industry.

We have several research questions in our sights:

– **Word disambiguation in context:** current methods represent a word in the vector space as the centroid of its different meanings. We want to evaluate an approach based on random walks through texts so as to distinguish the representation of words in context.
– **Compositionality of segment representation:** semantic vectors representing the meaning of segments should reflect compositionality of meaning of its parts, e.g. words, phrases and sentences.
– **Representation of narrativity:** we may represent narrative text qualities [5] as a *trajectory* of words or nuggets in vector space, e.g. document representation may be a trajectory instead of a point.

Acknowledgments. The authors thank all proofreaders for their detailed commentary and suggestions for improvements. Funding by TA ČR Omega grant TD03000295 is gratefully acknowledged.

References

1. Blei, D.M.: Probabilistic topic models. Commun. ACM 55(4), 77–84 (Apr 2012), http://doi.acm.org/10.1145/2133806.2133826
2. Buckley, C., Voorhees, E.M.: Retrieval evaluation with incomplete information. In: Proc. of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 25–32. SIGIR '04, ACM, New York, NY, USA (2004), http://doi.acm.org/10.1145/1008992.1009000
3. Cormack, G.V., Grossman, M.R., Hedin, B., Oard, D.W.: Overview of the TREC 2010 legal track. In: Proc. 19th Text REtrieval Conference. pp. 1–45. National Institute of Standards and Technology, Gaitherburg, MD (2010), http://trec.nist.gov/pubs/trec19/papers/LEGAL10.OVERVIEW.pdf
4. Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.A.: Indexing by latent semantic analysis. Journal of the American Society of Information Science 41(6), 391–407 (1990)
5. Hoenkamp, E., Bruza, P., Song, D., Huang, Q.: An Effective Approach to Verbose Queries Using a Limited Dependencies Language Model. In: Azzopardi, L., Kazai, G., Robertson, S.E., Rüger, S.M., Shokouhi, M., Song, D., Yilmaz, E. (eds.) ICTIR. Lecture Notes in Computer Science, vol. 5766, pp. 116–127. Springer (2009), http://dx.doi.org/10.1007/978-3-642-04417-5_11
6. Kdorff: Bpreftreceval2006 (2007), http://icb.med.cornell.edu/wiki/index.php/BPrefTrecEval2006, Accessed: 2016-10-29
7. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality (2013), http://arxiv.org/abs/1310.4546, arXiv:1310.4546
8. 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. Valletta, Malta (2010), software available at http://nlp.fi.muni.cz/projekty/gensim
9. TREC version of EDRM Enron Dataset, version 2, Accessed: 2016-10-29, http://www.edrm.net/resources/data-sets/edrm-enron-email-data-set-v2
10. Wang, L., Tasoulis, S., Roos, T., Kangasharju, J.: Kvasir: Seamless integration of latent semantic analysis-based content provision into web browsing. In: Proc. of the 24th International Conference on World Wide Web. pp. 251–254. WWW '15 Companion, ACM, New York, NY, USA (2015), http://doi.acm.org/10.1145/2740908.2742825

Part III

Electronic Lexicography and Language Resources

Automatic Identification of Valency Frames in Free Text

Martin Wörgötter

Natural Language Processing Centre
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic

Abstract. In this paper I present a versatile tool for automatic labelling of Czech verbs in free text with VerbaLex valency frames. The effective implementation can process one sentence in 0.03 seconds on average. I provide an overview of the algorithm and its evaluation.

Key words: VerbaLex, WordNet, valency frame, verb, annotation, tagger

1 Introduction

Valency lexicons are important lexical resources, which make it possible to disambiguate morphological, syntactic as well as semantic aspects of language. One such resource is *VerbaLex* [1] developed at the *Natural Language Processing Center at the Faculty of Informatics*. A significant feature of *VerbaLex* is its interconnection with the semantic lexical database *WordNet* [2].

Another notable verb valency lexicon for Czech is *VALLEX* [3]. A machine learning algorithm for matching verbs with corresponding valency frames of *VALLEX* was proposed in [4].

Assigning appropriate *VerbaLex* valency frames to verbs in a text is challenging. The difficulty of the task lies in discriminating between multiple valency frames. The algorithm for solving this task must necessarily encompass morphological and syntactic analysis.

In the following sections, I describe the implemented rule-based algorithm and the evaluation on five manually prepared test sets. Due to the exploitation of verb valency features specific to *VerbaLex*, a generalization of the algorithm for use with other verb valency lexicons is quite limited.

2 Implementation

The tool processes the output of the syntactic parser *SET* [5] given the options `-preserve-xml-tags` and `-long-phrases`. The second option ensures that morphological tagging is output together with the syntactic tree.

Syntactic analysis segments the input into clauses, which are the main scope for the algorithm. In each clause, each verb is looked up in *VerbaLex* for a list of possible candidate valency frames, i.e. those which allow the given lemmatized form of the verb. A set of tests, which is depending on the frame specification, is evaluated on each candidate. Only if all tests succeed, the candidate is accepted. Each verb triggers the following two tests.

- Principal verb—auxiliary verbs are discovered using the syntactic structure of the clause and are discarded.
- Reflexivity—the verb has to be reflexive/irreflexive according to the frame specification. This is verified by searching for a reflexive particle.

Valency frames in *VerbaLex* have the structure of a list of participants, which are either obligatory or facultative. Participants can capture semantic information via subcategorization features represented by *WordNet* literals. Surface grammar constraints are encoded using the properties listed bellow. Multiple possible values for each constraint are supported.

Depending on its specification, each obligatory participant imposes some of the following tests.

- Subcategorization features require a constituent which falls into the set of hyponyms of the second level semantic role.
- Surface grammar constraints are the following:
  - Case: same case number
  - Category of personality: a heuristic by which the grammatical gender of a *masculine noun* has to match the specified category
  - Prepositional lemma: it has to be found
  - Adverb: it has to match a word
  - Infinitive: a verb in infinitive has to be found
  - Subordinating conjunction lemma: has to be found in a subordinate clause

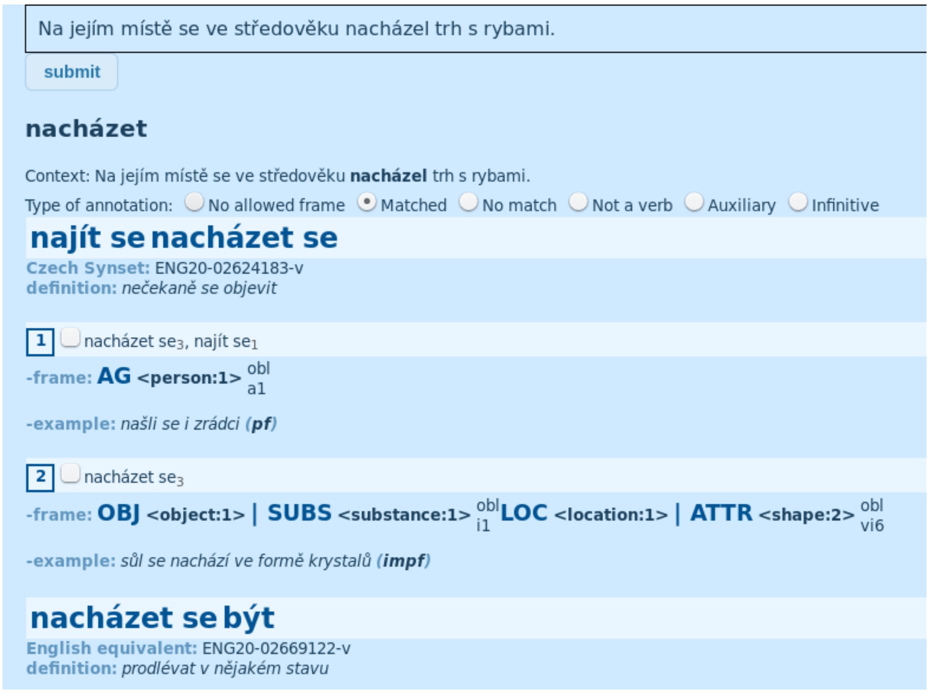
The described algorithm heavily relies on database lookups of *VerbaLex* and *WordNet*. To achieve the required performance for tagging large corpora, a command line option is available which employs a caching procedure during initialization to overcome this problem.

3 Preparation of a gold standard

Currently, no collection of sentences, manually annotated with *VerbaLex* valency frames is available, thus it was necessary to prepare the annotated data.

A simple web application, depicted in Figure 1 served this purpose. The annotators were five students with a specialization in computational linguistics.

The sentences for annotation were taken from the *czTenTen* [6] corpus using the *Sketch Engine* corpus interface [7] to filter out sentences not containing verbs and sort them by their *GDEX* score [8], a value expressing suitability for being used as a dictionary example. From the resulting list the top 900 sentences were extracted. 150 sentences were put aside and the rest was divided into five



Na jejím místě se ve středověku nacházel trh s rybami.

submit

nacházet

Context: Na jejím místě se ve středověku **nacházel** trh s rybami.

Type of annotation: No allowed frame [✓ Matched] No match Not a verb Auxiliary Infinitive

najít se nacházet se

Czech Synset: ENG20-02624183-v
definition: *nečekaně se objevit*

1 nacházet se$_3$, najít se$_1$

-frame: **AG** **<person:1>** $^{obl}_{a1}$

-example: *našli se i zrádci* (***pf***)

2 nacházet se$_3$

-frame: **OBJ** **<object:1>** | **SUBS** **<substance:1>** $^{obl}_{i1}$ **LOC** **<location:1>** | **ATTR** **<shape:2>** $^{obl}_{vi6}$

-example: *sůl se nachází ve formě krystalů* (***impf***)

nacházet se být

English equivalent: ENG20-02669122-v
definition: *prodlévat v nějakém stavu*

Fig. 1: The web application interface for annotating verbs.

sets. The 150 separate sentences were then randomly intermingled into each set raising their size to 300. The intersection between sets was later used to assess the inter-annotator agreement.

Each annotator was required to choose exactly one of the following options for each verb.

1. *No allowed frame*: a preselected option in case the verb is not recorded in *VerbaLex*
2. *Match*: a suitable frame was found
3. *No match*: no appropriate frame is available in *VerbaLex*
4. *Not a verb*: the word was erroneously recognized as a verb
5. *Auxiliary*: the verb is auxiliary
6. *Infinitive*: an infinite verb.

Only in case of a match, the annotator continued by choosing one or more appropriate valency frames which were listed with respect to the verb lemma.

4 Evaluation

The difficulty of the task is underlined by the obtained inter-annotator agreement. Only in 17.5%, all five annotators agreed on the same set of valency frames. To alleviate this problem, four gold standards, according to the number of agreements between annotators, were established, see Table 1.

For each gold standard the third column presents the total number of verbs for which at least 2-5 annotators (depending on the gold standard) agreed to assign at least one frame (corresponds to the option *match* described above). The last column shows the number of verbs for which the annotators agreed on the exact set of assigned frames, with respect to the third column.

The evaluation of the implementation is presented in Table 2. According to the results nearly every third analysed verb is correctly assigned the exact set of appropriate valency frames.

Table 1: Gold standards according to the number of agreements between annotators.

name	agreements	agreed to assign	full agreement (%)
GS1	at least 2	160	70.0
GS2	at least 3	119	43.7
GS3	at least 4	81	34.6
GS4	at least 5	40	17.5

Table 2: Evaluation results.

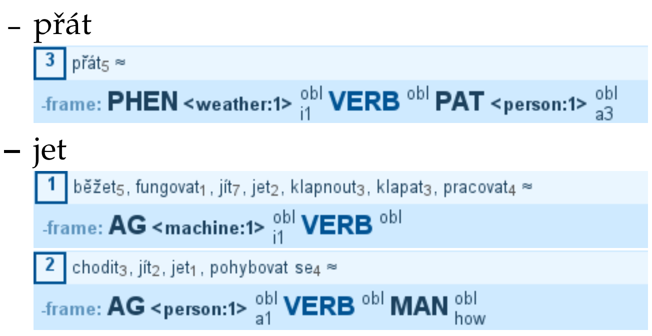
	precision (%)	recall (%)	F-score (%)
GS1	13.8	8.0	10.1
GS2	21.2	13.4	16.4
GS3	31.5	21.4	25.5
GS4	25.0	14.2	18.1

5 Error analysis

Figure 2 gives an example of both a successful and unsuccessful assignment of frames for verbs *přát* and *jet*.

The only frame accepted by the algorithm for verb *přát* is plausible. A problem occurs in the subordinate clause, as the *agens* does not match the semantic role *machine:1*. The algorithm assigned two frames, the first one being incorrect due to missing anaphora resolution.

The algorithm is very sensitive in processing the results of syntactical and morphological analysis and cannot cope with errors in the input data, which is



– přát

3 přát$_5$ ≈

-frame: PHEN <weather:1>$^{obl}_{i1}$ VERB^{obl} PAT <person:1>$^{obl}_{a3}$

– jet

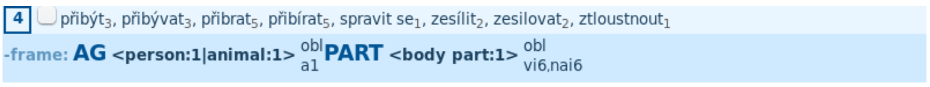
1 běžet$_5$, fungovat$_1$, jít$_7$, jet$_2$, klapnout$_3$, klapat$_3$, pracovat$_4$ ≈

-frame: AG <machine:1>$^{obl}_{i1}$ VERB^{obl}

2 chodit$_3$, jít$_2$, jet$_1$, pohybovat se$_4$ ≈

-frame: AG <person:1>$^{obl}_{a1}$ VERB^{obl} MAN$^{obl}_{how}$

Fig. 2: Example of erroneous result: "Počasí nám zatím nepřálo a tak jsme rychle jeli dál." (The weather was not good so far so we quickly went away.)



4 přibýt$_3$, přibývat$_3$, přibrat$_5$, přibírat$_5$, spravit se$_1$, zesílit$_2$, zesilovat$_2$, ztloustnout$_1$

-frame: AG <person:1|animal:1>$^{obl}_{a1}$ PART <body part:1>$^{obl}_{vi6,nai6}$

Fig. 3: Example of erroneous result: "Pokud se to ignoruje a pokračuje se v sestupu, bolest v uších stále zesiluje." (If it is ignored and the descent continues, the pain in the ears keeps increasing.)

demonstrated in Figure 3. An irrelevant frame is assigned to verb *zesílit* in the third clause. According to the syntactical analysis, this clause has a zero subject and the noun *bolest* is syntactically an object and any subcategorization features on a zero subject succeed.

6 Conclusions

In this paper I presented a tool, which is able to assign appropriate *VerbaLex* valency frames to verbs in free text.

The evaluation has proven the difficulty of the task, especially considering inter-annotator agreement. On the other hand, by relaxing the conditions for matching the gold standard, better results could be achieved.

The provided implementation could be used to enhance *VerbaLex* by semi-automatically adding corpus examples to the respective valency frames.

Acknowledgments. This work has been partly supported by the Ministry of Education of CR within the LINDAT-Clarin project LM2015071 and the Specific University Research.

References

1. Hlaváčková, D.: Databáze slovesných valenčních rámců VerbaLex. Disertační práce, Masarykova univerzita, Filozofická fakulta, Brno (2008)

2. Fellbaum, C.: WordNet: An Electronic Lexical Database. Language, speech, and communication. MIT Press, Cambridge (1998)
3. Žabokrtský, Z., Lopatková, M.: Valency information in VALLEX 2.0: Logical structure of the lexicon. The Prague Bulletin of Mathematical Linguistics (87) (2007) 41–60
4. Lopatková, M., Bojar, O., Semecký, J., Benešová, V., Žabokrtský, Z.: Valency lexicon of czech verbs vallex: Recent experiments with frame disambiguation. In: Text, Speech and Dialogue. Volume 3658., Karlovy Vary, Česká republika, Springer (2005) 99–106
5. 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, Springer (2011) 161–171
6. Jakubíček, M., Kilgarriff, A., Kovář, V., Rychlý, P., Suchomel, V.: The TenTen Corpus Family. In: 7th International Corpus Linguistics Conference CL. (2013) 125–127
7. 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)
8. Rychlý, P., Husák, M., Kilgarriff, A., Rundell, M., McAdam, K.: GDEX: Automatically finding good dictionary examples in a corpus. In: Proceedings of the XIII EURALEX International Congress, Barcelona, Institut Universitari de Lingüística Aplicada (2008) 425–432

Data Structures in Lexicography: from Trees to Graphs

Michal Měchura

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

Abstract. In lexicography, a dictionary entry is typically encoded in XML as a tree: a hierarchical data structure of parent-child relations where every element has at most one parent. This choice of data structure makes some aspects of the lexicographer's work unnecessarily difficult, from deciding where to place multi-word items to reversing an entire bilingual dictionary. This paper proposes that these and other notorious areas of difficulty can be made easier by remodelling dictionaries as graphs rather than trees. However, unlike other authors who have proposed a radical departure from tree structures and whose proposals have remained largely unimplemented, this paper proposes a conservative compromise in which existing tree structures become augmented with specific types of inter-entry relations designed to solve specific problems.

Key words: e-lexicography, dictionary writing systems, placement of multi-word items in dictionaries, bilingual dictionary reversal

1 A brief history of computerization in lexicography[1]

Following Atkins and Rundell [2, p. 3], there are three stages in the dictionary-writing process where computer software comes in: (1) as **corpus query systems** for discovering lexical knowledge in corpora, (2) as **dictionary writing systems** where lexical knowledge is encoded into a form suitable for presentation to human readers and (3) as **websites, apps** etc. which deliver the dictionary onto a user's screen. Together these three areas constitute the discipline known as *e-lexicography* (a good introduction to which is [5]).

[1] It is important to clarify that, in this paper, the term *lexicography* means writing dictionaries for humans: a discipline whose goal is not only to discover the properties of words (a goal it shares with lexicology) but also to communicate those discoveries successfully to human consumers who are neither lexicologists nor lexicographers: to "identify the most effective ways to present the linguistic properties of words in dictionaries according to specific criteria such as the type of dictionary, the intended user group, etc." [7, p. 4]. This separates human-oriented lexicography from computational lexicons such as WordNet [4].

ost innovation in e-lexicography has happened in (1) corpus query systems: so much, in fact, that corpus-driven methods have redefined dictionaries from intuition-based prescriptions to evidence-based descriptions. At the other end of the pipeline, in (3) dictionary publishing, websites and other electronic media had for long only imitated the behaviour of paper dictionaries. Lately, however, some innovation started appearing in this area as new methods of delivering dictionary content to users are emerging while dictionaries are becoming divorced from the original print medium, see e.g. [11].

The area where the least amount of innovation has happened until now is the middle part, (2) dictionary writing. Even though dictionary writing has become completely computerized in the last few decades, the structure of dictionaries we write today has not changed since pre-computer times. Yes, today's dictionary entries tend to be more easily navigable due to generous use of colour, font and whitespace, but that is only a superficial difference in formatting. Yes, today's dictionary writing software ensures that dictionary entries comply with a given schema, but this only replicates what lexicographers would be doing on paper or in a word processor anyway, only with more effort and less perfection. The underlying paradigm has not changed: a dictionary entry is still the same tree structure in which elements such as headwords, senses, part-of-speech labels and example sentences are stacked inside each other by means of parent-child relations where each child has at most one parent. The fact that we still model dictionary entries as trees means that some aspects of the lexicographer's work remain unnecessarily difficult.

2 What we can't do with dictionaries

Here we introduce two well-known problems in lexicography, each of which can be understood as an inconvenient consequence of the tree-like data structure dictionaries are encoded in.

2.1 Placement of multi-word items

Deciding under which headword a multi-word phraseme should be placed is a classical problem in lexicography [3]. Should an item like *third time lucky* be included under *third*, *time* or *lucky*? Arguably the best answer is 'all of them' but the only way to make it appear under all relevant headwords is by copying it. The traditional data structure of dictionary entries as trees imposes the inconvenient constraint that information cannot be shared across multiple entries (other than by copying). This difficulty can of course be worked around further downstream by clever search algorithms, by some form of indexing or cross-referencing, but it would be smarter to fix the problem at source by devising a data structure that allows fragments of entries to be 'shareable', able to appear in multiple entries. This is impossible in a tree structure where each phraseological element can have only one parent, but it is perfectly possible in a graph structure where it can have multiple parents, giving us a method to model many-to-many relations between entries and phrasemes.

2.2 Bilingual dictionary reversal

Another well-known problem in lexicography is reversing a bilingual dictionary [10]. Once we have written a bilingual dictionary from language X to language Y, it is far from trivial to convert it into a dictionary that goes in the opposite direction, from language Y to language X. There are points of indeterminacy which prevent us from doing it completely automatically. More importantly, the process is a one-way street: once we have reversed the dictionary, we have lost the connection between the source and the target: each entry in each dictionary is its own tree structure with no explicit links between them. If and when the source dictionary changes, the reversed dictionary has potentially become outdated as there is no automated way to project changes from one into the other. A more attractive proposition would be to encode pairs of bilingual dictionaries in a structure that keeps them synchronized, so that every element in every entry in the reversed dictionary 'knows' which element in which entry in the original dictionary it came from, and can react to changes. Again, this calls for a graph-based data structure where each element can have relations with other things besides its hierarchical parent.

3 Are graphs the answer?

While trees are the conventional data structure in human-oriented lexicography, lexicons for machines are often encoded as graphs. A typical example is WordNet [4] and other semantic networks which, in effect, are models of the mental lexicon. These seem like a promising source of inspiration. Instead of writing a tree-structured dictionary, one could build a graph-based model of the mental lexicon and then **derive** dictionaries from it, automatically and on demand. The conventional tree-structured entry would become a non-persistent output format, one of many possible 'views' of the graph, while problems such as multi-word item placement and dictionary reversal would disappear. In practice, however, all attempts to build a human-oriented dictionary in this way have so far remained experimental (e.g. [12]). It seems that the lexicography industry is not (yet?) prepared to 'think outside the tree' – or is perhaps the idea itself unrealistic because the lexical needs of humans and machines are incompatible?

Lately, some dictionary publishers have become inspired by the Semantic Web and started experimenting with re-encoding dictionaries as RDF graphs (e.g. [1], [8]). This is a more realistic attempt at innovation because, unlike semantic networks *à la* WordNet, it does not attempt to model the mental lexicon. Instead, it merely captures the same information dictionaries already have in trees and encodes it in a graph. In an RDF graph, dictionary entries can be augmented with various relations which 'break out' of the tree paradigm, for example sense-to-sense links between synonyms. The relations envisaged above, such as many-to-many relations between multi-word phrasemes and word senses, could be accommodated in an RDF graph easily. However, the disadvantage of RDF graphs (and graphs in general) is that they are not as easily human-readable as XML trees (and trees in general), not to mention human-writeable. Trees can be visualized neatly as two-dimensional objects, while graphs often can't. Trees are easy for humans to grasp mentally, while graphs are more difficult to 'take in'. For this reason, it is unlikely that lexicographers will switch to authoring graph-based dictionaries directly any time soon. All RDF encodings of human-oriented dictionaries have so far been automatic conversions from pre-existing tree-structured XML.

The problem then is that, while graphs are the more adequate structure for dictionaries, trees are more 'lexicographer-friendly'. What we need is a compromise: a set up which keeps dictionaries in a tree-like structure as much as possible, but which also allows them to 'break out' of the tree when necessary: for example to allow the sharing of phraseological subentries between entries. Importantly, we also need a dictionary writing system which allows lexicographers to work with dictionary entries in the familiar tree format as much as possible, while only forcing them to 'think outside the tree' when necessary.

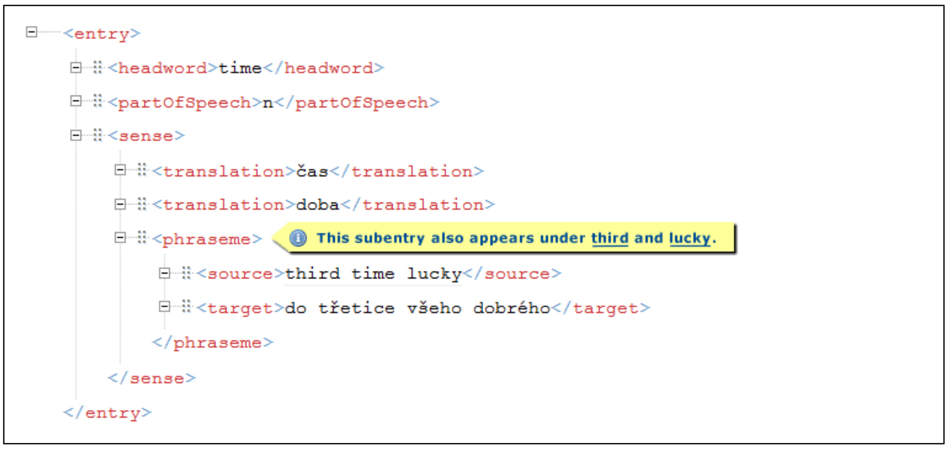
4 Introducing graph-augmented trees

In the model proposed in this paper, dictionaries will continue to be written in conventional tree-structured XML – or so they will appear to the lexicographers. Behind the scenes, the dictionary writing system will keep track of any relations that 'break out' of the tree and present them to the lexicographer as annotations beside the tree. The rest of this section will show how this approach will alleviate the two lexicographic problems outlined above.

4.1 Placement of multi-word items

An administrator will be able to specify in the dictionary schema which elements in the tree structure can be shared by multiple entries. This will typically apply to phraseological subentries and other multi-word material. When creating a phraseological subentry inside an entry, the lexicographer will be able to create new subentries as normal, but will also be able (and encouraged) to link to existing ones when applicable.

For example, a lexicographer will create the subentry *third time lucky* while working on the entry for *third*. To the lexicographer, it will seem as if the subentry is part of the entry, just like any other XML element. Internally, however, the system will store this subentry separately and link it to the entry for *third*. Later, while working on the entries for *time* and *lucky*, if the lexicographer decides to include *third time lucky* as subentry, he or she will be prompted by the system to bring in the existing subentry instead of creating a new one. Because the subentry is now shared by several entries, any changes made to it will affect all the entries that share it. When editing an entry that contains a shared subentry, the lexicographer will be notified (see Fig. 1) to



```
<entry>
  <headword>time</headword>
  <partOfSpeech>n</partOfSpeech>
  <sense>
    <translation>čas</translation>
    <translation>doba</translation>
    <phraseme>   This subentry also appears under third and lucky.
      <source>third time lucky</source>
      <target>do třetice všeho dobrého</target>
    </phraseme>
  </sense>
</entry>
```

Fig. 1: Notifying the lexicographer of relations that 'break out' of the tree: "This subentry also appears under 'third' and 'lucky'."

make sure they understand that any changes they make to the subentry here will be visible in the other entries too.

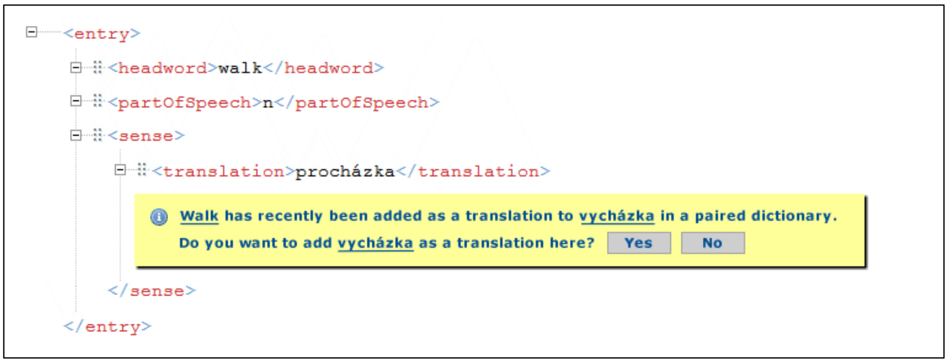
The model proposed here is similar to an approach one often sees in dictionaries where multi-word phrasemes are treated as independent entries, in effect promoting them to the same level as single-word entries. We may call this the 'multi-word promotion' approach. Multi-word promotion solves the problem of phraseme placement by deciding not to place the phraseme anywhere, and that is also its drawback: it strips the lexicographer of the ability to include a phraseme like *third time lucky* in a specific sense of a single-word entry, for example a specific sense of *time*.

The 'sharing' model proposed here is in fact an implementation of a less-known feature of Lexical Markup Framework (LMF) [6] where multi-word entries can be independent entries which can then be linked to from specific senses of other entries via their ID.

4.2 Bilingual dictionary reversal

An administrator will be able to set up a 'mapping' between the schemas of two dictionaries, such as a pair of dictionaries where one goes from language X to language Y and the other from language Y to language X. These dictionaries will then be 'paired'. As lexicographers make edits to entries in one of the dictionaries, the system will keep track of the edits and later suggest corresponding edits to the other dictionary in the pair. For example, when a lexicographer adds the translation *walk* under the headword *vycházka* in one dictionary, the system will remember to suggest adding the reverse translation *vycházka* to the appropriate headword *walk* in the other dictionary (see Fig. 2).

This way the lexicographers will be encouraged to keep the two dictionaries synchronized.



```
<entry>
  <headword>walk</headword>
  <partOfSpeech>n</partOfSpeech>
  <sense>
    <translation>procházka</translation>
      Walk has recently been added as a translation to vycházka in a paired dictionary.
      Do you want to add vycházka as a translation here?   Yes   No
  </sense>
</entry>
```

Fig. 2: Keeping paired dictionaries synchronized semi-automatically: "'Walk' has recently been added as a translation to 'vycházka' in a paired dictionary. Do you want to add 'vycházka' as a translation here? Yes – No."

The model proposed here is similar to, but subtly different from, the approach sometimes taken by dictionary projects where lexemes exist not as strings but as links to another database. For example, in the Cornetto project [9] there are two databases: a monolingual dictionary and a wordnet. The wordnet does not contain any literal lexemes: instead, it has links to specific senses of specific headwords in the monolingual dictionary. If headwords in the monolingual dictionary are changed or deleted, the changes will be refected in the wordnet automatically.

The 'pairing' model proposed here does not envisage such automation: it does not envisage that changes in one dictionary would be reflected in another dictionary automatically. Instead, the system would only keep track of changes in one place and **suggest** corresponding changes in other places. It would be up to the lexicographer to accept or reject the suggestions. The fact that the pairing is not fully automatic is what, it is hoped, would make this way of working more compatible with how lexicographers usually work: the final content of each and every entry would be the result of a lexicographer's decision, like it always has been in lexicography – except this time the decisions would be 'computer-aided' (consider the analogy of Computer-Aided Translation, CAT, where a software tool suggests candidate translations and a human translator either accepts or rejects them).

4.3 Other benefits of graph-augmented trees

The hybrid data model proposed here has benefits that stretch beyond the two scenarios described above.

The notion of shareable subentries can be used for other entry components besides phrasemes, such as example sentences. A sentence like *who's the lucky winner?* is a good illustrative example for both *lucky* and *winner*. Instead of creating two copies of the sentence in two entries, it could be stored in a single copy internally and **shared** by the entries. Later, if lexicographers want to edit the sentence (say to correct a spelling mistake) or add a translation to it, they only need to do it once, saving work and avoiding any potential for inconsistencies.

The same could even apply to translation equivalents inside senses. In many dictionaries translations are nothing more than strings of text but, in some, translations are decorated with extensive grammatical and other annotations. When the same translation appears under multiple headwords, as they often do, lexicographers' time is wasted entering the same information again and again. Instead, translations could be 'shareable', thus again saving work and avoiding potential inconsistencies.

The concept of paired dictionaries too can be used for other purposes besides bilingual reversal. The paired dictionaries can be related by means other than reversal, for example by the lexicographic function [13] they fulfil, such as the type of their target audience: one can be a beginner's dictionary and the other a larger dictionary for advanced learners of the same language. In such a situation, an entry in the beginner's dictionary is typically an abridged version of its counterpart in the advanced dictionary. When a lexicographer makes an edit to one of the pair, such as add a new translation or an example sentence, the system will remember to propose a corresponding edit in the other dictionary, thus helping to keep the two synchronized.

The notions of 'sharing' and 'pairing' can even be combined into a single setup. For example, a dictionary and a thesaurus of the same language can share definitions, while the system keeps track of paired senses in both.

5 Conclusion

As an industry, lexicography is facing life-changing challenges at the moment. As revenue from commercial dictionary sales is decreasing, lexicography is moving from the private sector to the public sector where it needs to function on limited budgets. In such circumstances it becomes important to be able to 'do more with less': to deliver more dictionaries more quickly, with less effort. The model of graph-augmented trees, if and when it becomes implemented in an industrial-strength dictionary writing system, will empower lexicographic teams to deliver precisely that, while allowing them to continue working within the familiar paradigm of trees. The techniques of 'sharing' and 'pairing' are time-saving devices which remove the need for repetitive data entry and simultaneously ensure greater consistency between individual entries and entire dictionaries.

Acknowledgments. This publication was written with the support of the Specific University Research provided by the Ministry of Education, Youth and Sports of the Czech Republic.

References

1. Aguado-de-Cea, G., Montiel-Ponsoda, E., Kernerman, I., Ordan, N. (2016). From dictionaries to cross-lingual lexical resources. In: Kernerman Dictionary News, 24, pp. 25-31.
2. Atkins, B. T. S., Rundell, M. (2008). The Oxford guide to practical lexicography. Oxford: Oxford University Press.
3. Bogaards, P. (1990). Où cherche-t-on dans le dictionnaire ? In: International Journal of Lexicography, 3(2), pp. 79-102.
4. Fellbaum, C. (1998). WordNet: an electronic lexical database. Cambridge: MIT Press.
5. Granger, S., Paquot, M. (2012). Electronic lexicography. Oxford: Oxford University Press.
6. ISO 24613:2008 Language resource management – Lexical markup framework (LMF). `http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=37327`
7. Ježek, E. (2016). The lexicon: an introduction. New York: Oxford University Press.
8. Klimek, B., Brümmer, M. (2015). Enhancing lexicography with semantic language databases. In: Kernerman Dictionary News, 23, pp. 5-10.
9. Horák, A., Rambousek, A., Vossen, P., Segers, R., Maks, I., van der Vliet, H. (2009) Cornetto Tools and Methodology for Interlinking Lexical Units, Synsets and Ontology. In: Current Issues in Unity and Diversity of Languages. Seoul: The Linguistic Society of Korea, pp. 2695-2713.
10. Maks, I. (2007). OMBI: The Practice of Reversing Dictionaries. In: International Journal of Lexicography, 20(3), pp. 259-274.
11. Měchura, M. (2016). Things to think about when building a dictionary website. Talk at meeting of European Network of e-Lexicography. `http://www.lexiconista.com/things-to-think.pdf`
12. Polguère, A. (2004). From Writing Dictionaries to Weaving Lexical Networks. In: International Journal of Lexicography, 24(7), pp. 396-418.
13. Tarp, S. (2008). Lexicography in the Borderland Between Knowledge and Non-Knowledge: General Lexicographical Theory with Particular Focus on Learner's Lexicography. Tübingen: De Gruyter.

Pre-processing Large Resources for Family Names Research

Adam Rambousek

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

Abstract. This paper describes methodology and tools used to pre-process historical archive documents in various formats and their conversion to unified format. Resources were used to investigate the origins and geographical distribution of surnames in the United Kingdom, as part of the Family Names in Britain and Ireland research project. Data extracted from the documents and their connection proved to be valuable research resource which helped to speed up the lexicographic work.

Key words: DEB platform, lexicography, big data, family names, data conversion

1 Introduction

Family Names in Britain and Ireland (FaNBI) [1] is the research project of the University of the West of England that started in 2010, the first phase finished successfully in May 2014 and the research was extended to 2016.

FaNBI aims to complete a detailed investigation of the origins, history, and geographical distribution of the 45,000 most frequent surnames in the United Kingdom. The DEB platform [2,3], developed at the NLP Centre FI MU, was selected by the University of the West of England (UWE) as the dictionary writing system for the project because of its versatility and possibility to combine various resources. This paper describes the tools and methods used to pre-process various resources needed for the research.

2 List of names and frequencies

The list of frequency for each family name is the cornerstone of the FaNBI project. It is not only the list of entries to edit, but the frequency also decides which names will be edited in each phase of the project. In the first phase, all names with more than 100 bearers were edited. The work was extended to all names with more than 20 bearers in the second phase.

At the beginning of the project, two lists were used – 1881 census report [4] and 1997 statistical data [5]. However, both lists had to be preprocessed and filtered, because they contained a lot of noise and errors (for example, spelling errors or invalid characters). Another issue with the lists provided was that all the names were written in uppercase. A straightforward solution is to leave the first letter of each word uppercase and the rest in lowercase, however, this is not true for all names. For example, Scottish and Irish names like O'Brian or McGaffin had to be considered. This type of names also produced the issue with various written spellings used. For the Mc- names, three different variant spellings were present – Mac-, Mc- and M'-. Similarly for O'- names, various apostrophe characters were used and sometimes the name was written without the apostrophe. It was decided to include only the spellings Mc- and O'- into the dictionary, and redirect readers searching for other variants to the correct dictionary entry. To make the matter even more complicated, some family names starting with the string Mac- are separate names and not the variant spellings of Mc-, for example Mach or Mackarel. To solve this issue, the list was edited in two steps. In the first step, variant spellings were detected and uncertain samples were reported. In the next step, the proposed changes were approved by the lexicographers. In case of variant spellings, the frequencies had to be summed for all the forms.

The 1881 census list was edited with the described method, and the method was updated for other lists. Lexicographers' approval was not needed anymore, because the 1881 list was included in the cleaning tool to decide the correct spellings and variant combination. Finally, only names with frequency of at least 20 were included. During the cleaning and combining of the 1881 census list, the number of records was reduced from 469,356 to 373,319 records.

With the report from recent years, the issue linked with growing immigration was discovered – both masculine and feminine forms of the names appeared for languages where these forms differ (for example, Polish or Czech). It was decided to keep only the masculine form of the family name, and the method for frequency inclusion was updated. Feminine forms are detected by the known suffix and if the masculine form is present in the database, the frequency numbers are combined.

3 Combining resources

In the aim to include as much historical evidence as possible, various existing databases are used to search for the records of the family names. A selection of records is available as a webservice from The National Archive[1], however it was needed to clean or preprocess the resources.

Very valuable resource for family names studies is the *International Genealogical Index* (IGI) [6] compiled by The Church of Jesus Christ of Latter-day Saints. The IGI contains worldwide records extracted from the parish archives and similar sources, or submitted by the members of the Church. IGI records are published on the FamilySearch website[2], however the website does not pro-

[1] `http://discovery.nationalarchives.gov.uk/`

[2] `https://familysearch.org/`

vided access to the complete collection and records may contain various errors or inconsistencies. The original database records for the Great Britain were provided to the FaNBI project. The database was transcribed from the parish archives by volunteers over the course of several decades. Because of many reasons (for example, unreadable books, different spellings by each transcriber, spelling mistakes etc.) the database had to be cleaned up before it could be included in the FaNBI research. Sometimes, several volunteers transcribed the same parish records, so the duplicate data had to be detected. The following list sums the process of the cleaning and deduplicating the IGI database.

– Original database contained 188,043,185 records. Each record contains information about the event type (birth, christening, marriage, or death), first name, surname, date, location (county, town/place name, sometimes the exact parish), and the role of the person (e.g. for marriage bride, groom, or their parents).
– Obvious mistakes were deleted, for example records claiming that the English cities are in France.
– Names of the counties were standardized from variant spellings and abbreviations.
– For each county, a list of place names was extracted. These lists were distributed amongst the volunteers from the Guild of One-Name Studies[3]. Volunteers checked if the place name on the list belongs to the given county, or provided correct spelling. As a result of this process, a standardized list of place names was created and the database records were fixed. The records that provided incorrect information about the place name were deleted.
– In the next step, duplicate records were deleted. Because the main aim for the FaNBI research was not to build complete and perfect database, but provide reliable evidence, it was possible to delete not just exact duplicates, but also suspect duplicates. The rules for duplicate detection were considering following information from the records: first name, surname, date, town, county, and event type. Records were flagged as duplicate when all information were identical, but one of the following fields was different: first name, town, county, or event type.
– At the end of the process, IGI database contained 72,187,630 records.

Subsequently, the database was used to automatically add historical evidence to the FaNBI dictionary. For each family name, IGI records were extracted for each century and most prominent county, formatted according to the reference templates and saved in the entry. 40,274 family names entries were automatically enhanced with the IGI evidence. Apart from the enhancement of the dictionary, the processed IGI database is regularly consulted by the researchers as a valuable resource. For the sample of original IGI record and converted form to include as the historical evidence see Table 1.

[3] `http://one-name.org/`

Table 1: Original record from the IGI database and form included into FaNBI.

Original record (batch identification, event date, event place, event type, year, first name, surname, role, gender):
Bletsoe, Bedford, England | 05 Sep 1629 | Bletsoe, Bedford, England | Christening | 1629 | John | Darter | Principal's Father | Male

Converted record:
John <sn>Darter</sn>, 1629 in <src>IGI</src> (Bletsoe, Beds)

Another archive resource that required preprocessing were three volumes of *The Irish Fiants of the Tudor sovereigns during the reigns of Henry VIII, Edward VI, Philip & Mary, and Elizabeth I* [7]. The Fiants contain various court warrants and are available in the electronic format. Each record is clearly marked in the text and thanks to the official language, it is possible to detect persons' names, occupations, or residence. In the first step, the Word documents (results of the OCR recognition) were converted to the XML format. Each court record was converted into a separate XML entry with enhanced metadata. For example, the date of the record was converted from the regnal years system to calendar years.

Converted XML documents were later processed by the extraction tool. The tool standardized common OCR misspellings and detected frequently repeating text patterns in the warrant texts. The list of place names (created during the IGI database cleanup) was used to detect town names and match them with the correct county. Where available, also the persons' occupations were tagged in the record. Finally, all the information were formatted according to the FaNBI reference templates and are available for reference in appropriate entries. For the sample of the conversion from Fiants to FaNBI, see Table 2.

Table 2: Original Fiants record and form converted to include in FaNBI.

Original record:
1431. Pardon to Thomas Dowdall, of Dermondston, county Dublin, husbandman.—2 November, xi.

Converted record:
Thomas <sn>Dowdall</sn>, 1569 in <src>Fiants Eliz</src> $1431 (Dermondston, co. Dublin)

4 Conclusions

We have presented methodology to extract valuable information from various resources, unify the data from several documents, and combine the data for lexicographic research. Dictionary based on the results of the FaNBI projects are scheduled for publication by the Oxford University Press on November 17, 2016 [4].

Combining various historical documents for a single family name into one dictionary entry helped to speed up the research and discover new connections in the data. Researchers and general public users also have the possibility to view much richer information in one place.

Proven methodology and tools from the FaNBI were later adapted for the creation of the Dictionary of American Family Names (2nd edition), starting in 2014 and aimed to be published by the Oxford University Press in 2017.

Acknowledgments. This work has been partly supported by the Ministry of Education of CR within the national COST-CZ project LD15066.

References

1. Hanks, P., Coates, R., McClure, P.: Methods for Studying the Origins and History of Family Names in Britain. In: Facts and Findings on Personal Names: Some European Examples, Uppsala, Acta Academiae Regiae Scientiarum Upsaliensis (2011) 37–58
2. Rambousek, A., Horák, A.: DEBWrite: Free Customizable Web-based Dictionary Writing System. In Kosem, I., Jakubíček, M., Kallas, J., Krek, S., eds.: Electronic lexicography in the 21st century: linking lexical data in the digital age. Proceedings of the eLex 2015 conference, Trojina, Institute for Applied Slovene Studies/Lexical Computing Ltd. (2015) 443–451
3. Horák, A., Rambousek, A.: Lexicographic tools to build new encyclopaedia of the czech. The Prague Bulletin of Mathematical Linguistics (106) (2016) 205–213
4. The National Archives: Census records, 1881 `http://www.nationalarchives.gov.uk/records/census-records.htm`.
5. Hanks, P., Coates, R.: Onomastic lexicography. In Fjeld, R.V., Torjusen, J.M., eds.: Proceedings of the 15th EURALEX International Congress, Oslo,Norway, Department of Linguistics and Scandinavian Studies, University of Oslo (aug 2012) 811–815
6. The Church of Jesus Christ of Latter-Day Saints: International Genealogical Index `https://www.familysearch.org/search/collection/igi`.
7. Nicholls, K., ed.: The Irish Fiants of the Tudor Sovereigns during the Reigns of Henry VIII, Edward VI, Philip and Mary, and Elizabeth I. Edmund Burke Publisher (1994)

[4] `https://global.oup.com/academic/product/the-oxford-dictionary-of-family-names-in-britain-and-ireland-9780199677764`

Options for Automatic Creation of Dictionary Definitions from Corpora

Marie Stará, Vojtěch Kovář

Faculty of Arts
and
Natural Language Processing Centre, Faculty of Informatics

Masaryk University
Brno, Czech Republic
413827@mail.muni.cz, xkovar3@fi.muni.cz

Abstract. This paper maps the possibilities of using existing corpus tools to acquire definitions for Czech in an automatic way. It compares definitions from Dictionary of contemporary Czech (Slovník současné češtiny pro školu a veřejnost) and data acquired using Thesaurus and Word sketch in corpus czTenTen12.

Key words: dictionary definition, corpora, word sketch

1 Introduction

Creation of definitions is one of the key steps in compilation of a monolingual learner's dictionary.

There is a long tradition in creating learner's dictionaries with heavy support of corpora and related tools (corpora were first used in lexicography in Collins COBUILD English Language Dictionary published in 1987 [1]). However, so far there is no good automatic method of supporting lexicographers in creating definitions.

There have been attempts at automatic extraction of definitions from corpora [2,3,4,5,6], but it seems that there is simply not enough definitions in general language corpora. However, there may be enough information to *create* a definition, i.e. to aggregate the available information and build a new definition.

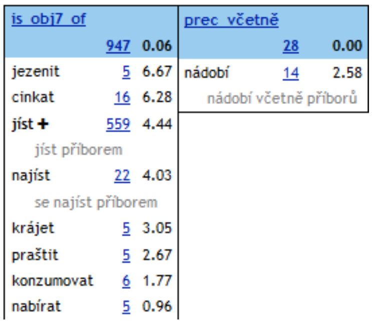
The purpose of this paper is to make a survey over the corpora data for Czech to find out, to what extent it contains information suitable for such automatic definition building.

2 Method

To create a definition we first need to know what a definition should contain. According to Manuál lexikografie (Manual of Lexicography) [7] the basic types of definitions are:

Table 1: Sketch Engine thesaurus for *příbor*

Lemma	Translation	Score	Freq
nádobí	utensils, tableware	0.209	74,734
vidlička	fork	0.195	15,428
talířek	dessert plate	0.167	8,831
tácek	coaster	0.159	6,615
tác	tray	0.150	11,233
talíř	plate	0.148	73,763
hrníček	cup	0.147	13,490
hrnek	mug	0.143	31,018
hrneček	cup	0.143	14,385
lžička	teaspoon	0.138	49,997



is_obj7_of	947	0.06
jezenit	5	6.67
cinkat	16	6.28
jíst +	559	4.44
jíst příborem		
najíst	22	4.03
se najíst příborem		
krájet	5	3.05
praštit	5	2.67
konzumovat	6	1.77
nabírat	5	0.96

prec_včetně	28	0.00
nádobí	14	2.58
nádobí včetně příborů		

Fig. 1: Word sketch for *příbor*

– intensional – traditional definition using genus and differentia or a list of subsets
– extensional – by listing every member of a set or using ostensive definition (defining by pointing)

Another possibility is to use a synonym or antonym.

Next, we need to find out if there is a way how to get such definitions (or something close to them) using current corpus tools. In the next sections, we show definitions from “Slovník spisovné češtiny pro školu a veřejnost” (Dictionary of contemporary Czech, further referred to as SSČ) [8], the latest Czech monolingual learner’s dictionary, and the data acquired from the 4-billion Czech web corpus `czTenTen12`, for a set of 10 words: nouns “příbor” (cutlery), “pes” (dog) and “bagr” (excavator); verbs “trpět” (suffer) and “chytit” (catch); adjectives “zádumčivý” (broody), “starý” (old) and “opilý” (drunk); conjugations “poněvadž” (because) and “nebo” (or), as an example of synsemantics.

This comparison forms the core of this paper and should show us if there is a potential for automatic creation of definitions from corpora.

3 Nouns

3.1 Příbor (cutlery)

According to SSČ, *příbor* has two meanings:

1. souprava náčiní, kterým se jí (lžíce, vidlička, nůž) (utensils used for eating (spoon, fork, knife))
2. souprava jídelního nádobí (set of eating utensils)

These two senses are practically identical and any automatic disambiguation would probably not be able to distinguish between them. However, according to Kilgarriff [9],

> if the instance exemplifies a pattern of use which is sufficiently frequent, and is insufficiently predictable from other meanings or uses of word, then the pattern qualifies for treatment as a dictionary sense.

SSČ does not abide this rule and lists results predictable from other meanings; so it is rather a problem in SSČ.

Table 1 shows that the word *nádobí* (utensils) is most similar to *příbor*. Word sketch shows that *příbor* is used together with *jíst* (to eat). Also from category *prec_včetně* (including), it is apparent that the word *nádobí* is a hypernym of *příbor*.

3.2 Pes (dog)

Pes has three meanings defined in SSČ:

1. šelma ochočená k hlídání, lovu ap. (domesticated carnivore for guarding, hunting etc.)
2. samec psovité šelmy (male canine)
3. expr. bezohledný, krutý člověk (expressively cruel person)

Table 2 shows that the word *zvíře* (animal) is most similar to *pes*. It is also a hypernym, however, this information is not present in the corpus results. Word sketch shows strong collocation between noun *pes* and verb *štěkat* (bark) – see Figure 2.

3.3 Bagr (excavator)

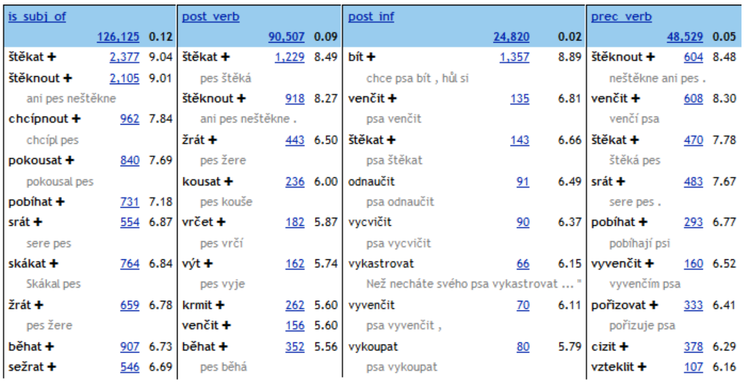
There are two meanings for the word *bagr* in SSČ:

1. rýpadlo (digger)
2. plavidlo k bagrování (dredge)

Table 3 shows that the word *bagr* is most similar to *rypadlo*. This is not a hypernym as in previous cases, but a synonym. The importance of this interrelation is described bellow. Word sketches provide little relevant information. There is *rypadlo* in category *coord* but it has low frequency and also a relatively low score. The other categories shown in Figure 3 are also not very useful. The verb with highest score is *zakousnout* (have a bite) which is far from perfect for dictionary definition (but still, it is relevant, as the other verbs in the lists).

Table 2: Thesaurus results for *pes*

Lemma	Translation	Score	Freq
zvíře	animal	0.468	564,849
kočka	cat	0.447	294,032
dítě	child	0.428	4,455,634
pejsek	doggy	0.423	174,852
kůň	horse	0.402	485,617
muž	man	0.380	1,616,460
člověk	human	0.378	8,036,909
žena	woman	0.370	1,899,472
kluk	boy	0.348	671,754
rodič	parent	0.324	949,735



is subj of			post verb			post inf			prec verb		
	126,125	0.12		90,507	0.09		24,820	0.02		48,529	0.05
štěkat +	2,377	9.04	štěkat +	1,229	8.49	bít +	1,357	8.89	štěknout +	604	8.48
štěknout +	2,105	9.01	pes štěká			chce psa bít , hůl si			neštěkne ani pes .		
ani pes neštěkne			štěknout +	918	8.27	venčit +	135	6.81	venčit +	608	8.30
chcípnout +	962	7.84	ani pes neštěkne .			psa venčit			venčí psa		
chcípl pes			žrát +	443	6.50	štěkat +	143	6.66	štěkat +	470	7.78
pokousat +	840	7.69	pes žere			psa štěkat			štěká pes		
pokousal pes			kousat +	236	6.00	odnaučit	91	6.49	srát +	483	7.67
pobíhat +	731	7.18	pes kouše			psa odnaučit			sere pes .		
srát +	554	6.87	vrčet +	182	5.87	vycvičit	90	6.37	pobíhat +	293	6.77
sere pes			pes vrčí			psa vycvičit			pobíhají psi		
skákat +	764	6.84	výt +	162	5.74	vykastrovat	66	6.15	vyvenčit +	160	6.52
Skákal pes			pes vyje			Než necháte svého psa vykastrovat ... "			vyvenčím psa		
žrát +	659	6.78	krmit +	262	5.60	vyvenčit	70	6.11	pořizovat +	333	6.41
pes žere			venčit +	156	5.60	psa vyvenčit ,			pořizuje psa		
běhat +	907	6.73	běhat +	352	5.56	vykoupat	80	5.79	cizit +	378	6.29
sežrat +	546	6.69	pes běhá			psa vykoupat			vzteklit +	107	6.16

Fig. 2: Word sketches for the word *pes*

3.4 Nouns: Summary

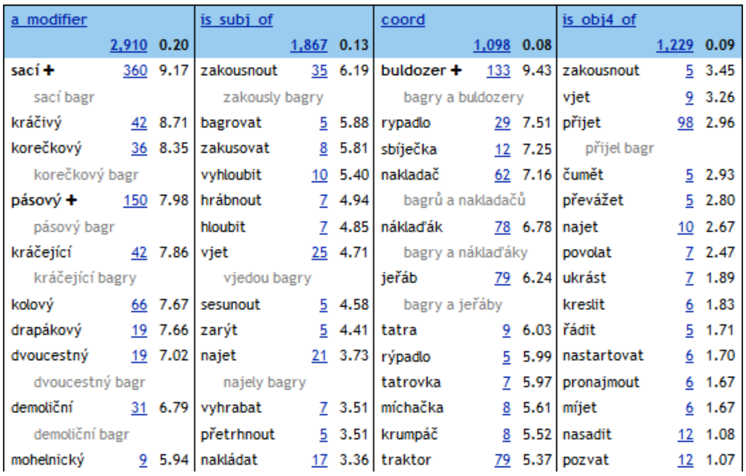
Some nouns can be defined as "*hypernym-from-thesaurus* that *verb-from-word-sketches*":

– příbor: nádobí, kterým se jí (cutlery: utensils that are used for eating)
– pes: zvíře, které štěká (dog: animal that barks)

However, this definition includes only the primary meaning. And it is not applicable for every noun, for example *bagr* is not really *rypadlo, které zakusuje* (*excavator* is not really *a digger that bites*). Also, it might be useful to distinguish when such hypernym is subject and when it is object (here distinguished by using passive voice) – current word sketch relations for Czech are not really good in this aspect.

Table 3: Thesaurus results for *bagr*

Lemma	Translation	Score	Freq
rypadlo	digger	0.244	3,452
buldozer	bulldozer	0.225	6,743
náklaďák	lorry	0.182	23,706
nakladač	traxcavator	0.179	10,753
rýpadlo	digger	0.159	1,263
jeřáb	derrick	0.146	31,472
traktor	tractor	0.141	63,830
kamión	lorry	0.131	11,206
kamion	lorry	0.127	64,591
tahač	tractor unit	0.115	11,959



a modifier			is subj of			coord			is obj4 of		
	2,910	0.20		1,867	0.13		1,098	0.08		1,229	0.09
sací +	360	9.17	zakousnout	35	6.19	buldozer +	133	9.43	zakousnout	5	3.45
sací bagr			zakously bagry			bagry a buldozery			vjet	9	3.26
kráčivý	42	8.71	bagrovat	5	5.88	rypadlo	29	7.51	přijet	98	2.96
korečkový	36	8.35	zakusovat	8	5.81	sbíječka	12	7.25	přijel bagr		
korečkový bagr			vyhloubit	10	5.40	nakladač	62	7.16	čumět	5	2.93
pásový +	150	7.98	hrábnout	7	4.94	bagrů a nakladačů			převážet	5	2.80
pásový bagr			hloubit	7	4.85	náklaďák	78	6.78	najet	10	2.67
kráčející	42	7.86	vjet	25	4.71	bagry a náklaďáky			povolat	7	2.47
kráčející bagry			vjedou bagry			jeřáb	79	6.24	ukrást	7	1.89
kolový	66	7.67	sesunout	5	4.58	bagry a jeřáby			kreslit	6	1.83
drapákový	19	7.66	zarýt	5	4.41	tatra	9	6.03	řádit	5	1.71
dvoucestný	19	7.02	najet	21	3.73	rýpadlo	5	5.99	nastartovat	6	1.70
dvoucestný bagr			najely bagry			tatrovka	7	5.97	pronajmout	6	1.67
demoliční	31	6.79	vyhrabat	7	3.51	míchačka	8	5.61	míjet	6	1.67
demoliční bagr			přetrhnout	5	3.51	krumpáč	8	5.52	nasadit	12	1.08
mohelnický	9	5.94	nakládat	17	3.36	traktor	79	5.37	pozvat	12	1.07

Fig. 3: Word sketches for the word *bagr*

4 Verbs

4.1 Trpět (to suffer)

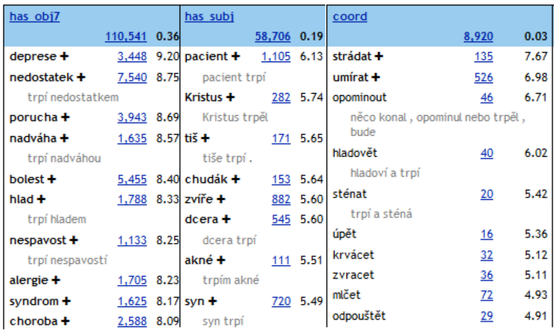
Trpět has four meanings defined in SSČ:

1. prožívat, snášet bolest, trýzeň, nepříjemnost (experience, bear pain, suffering, inconvenience
2. být nemocen n. jinak strádat (be ill or suffer)
3. (trpně) snášet (to bear patiently)
4. hovor. mít v oblibě, potrpět si (to like sth)

Table 4 shows that the word *trpět* is most similar to *projevovat* (to show) and *umírat* (to die). However, these are not hypernyms nor synonyms of *trpět*, although they are somehow semantically similar. Apparently we cannot define verbs in the same way as we outlined for nouns. Word sketch category *coord*

Table 4: Thesaurus results for *trpět*

Lemma	Translation	Score	Freq
projevovat	to show	0.255	221,222
umírat	to be dying	0.254	111,868
zemřít	to die	0.240	397,387
onemocnět	to fell ill	0.238	47,770
trápit	to afflict	0.225	237,852
cítit	to feel	0.219	970,162
žít	to live	0.214	1,333,268
umřít	to die	0.213	115,718
projevit	to show	0.211	326,269
způsobit	to cause	0.206	405,320



has_obj7	110,541	0.36
deprese +	3,448	9.20
nedostatek +	7,540	8.75
trpí nedostatkem		
porucha +	3,943	8.69
nadváha +	1,635	8.57
trpí nadváhou		
bolest +	5,455	8.40
hlad +	1,788	8.33
trpí hladem		
nespavost +	1,133	8.25
trpí nespavostí		
alergie +	1,705	8.23
syndrom +	1,625	8.17
choroba +	2,588	8.09

has_subj	58,706	0.19
pacient +	1,105	6.13
pacient trpí		
Kristus +	282	5.74
Kristus trpěl		
tiš +	171	5.65
tiše trpí .		
chudák +	153	5.64
zvíře +	882	5.60
dcera +	545	5.60
dcera trpí		
akné +	111	5.51
trpím akné		
syn +	720	5.49
syn trpí		

coord	8,920	0.03
strádat +	135	7.67
umírat +	526	6.98
opominout	46	6.71
něco konal , opominul nebo trpěl , bude		
hladovět	40	6.02
hladoví a trpí		
sténat	20	5.42
trpí a sténá		
úpět	16	5.36
krvácet	32	5.12
zvracet	36	5.11
mlčet	72	4.93
odpouštět	29	4.91

Fig. 4: Word sketches for the word *trpět*

shown in Figure 4 yields *strádat* (suffer), a synonym used in meaning 2 in SSČ. Categories *has_subj* and *has_obj7* show very relevant pattern of usage, e.g. *pacient trpí depresemi* (patient suffers by depression).

4.2 Chytit (to catch)

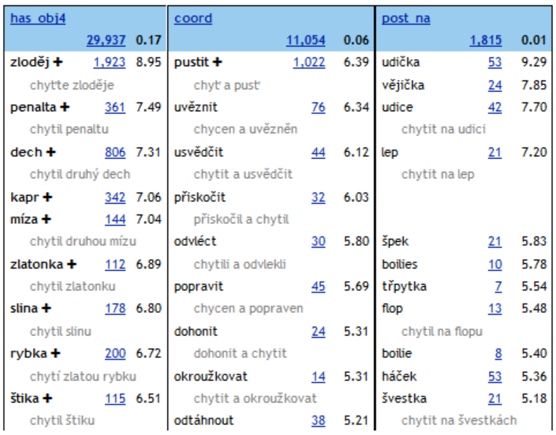
Chytit has, according to SSČ, seven meanings:

1. rukou n. rukama uchopit a podržet, prudce vzít (to grab sth by hand)
2. zmocnit se lovem ap. (to hunt down)
3. rychlým pohybem dostihnout (to catch)
4. hovor. dostat, získat (to gain)
5. zachytit se, přilnout (to hold on sth)
6. hovor. i chytit se, zachvátit, zmocnit se (to capture)
7. začít hořet, vzplanout (to catch fire)

Table 5 shows that the word *chytit* is most similar to *chytnout* and *chytat*. These verbs are not suitable for definition: *chytit* and *chytnout* are semantically identical and only their written form differs, *chytit* and *chytat* differ only in verbal aspect.

Table 5: Thesaurus results for *chytit*

Lemma	Translation	Score	Freq
chytnout	to catch	0.517	126,197
chytat	to catch	0.389	111,670
vytáhnout	to pull up	0.289	243,591
popadnout	to grab	0.278	43,277
pustit	to drop, to let go	0.276	490,512
vzít	to take	0.265	1,263,663
držet	to hold on	0.262	959,864
uchopit	to catch	0.251	44,547
zabít	to kill	0.246	339,422
uvidět	to see	0.235	707,341



has_obj4			coord			post_na		
	29,937	0.17		11,054	0.06		1,815	0.01
zloděj +	1,923	8.95	pustit +	1,022	6.39	udička	53	9.29
chyťte zloděje			chyť a pusť			vějička	24	7.85
penalta +	361	7.49	uvěznit	76	6.34	udice	42	7.70
chytil penaltu			chycen a uvězněn			chytit na udici		
dech +	806	7.31	usvědčit	44	6.12	lep	21	7.20
chytil druhý dech			chytit a usvědčit			chytit na lep		
kapr +	342	7.06	přiskočit	32	6.03			
míza +	144	7.04	přiskočil a chytil					
chytil druhou mízu			odvléct	30	5.80	špek	21	5.83
zlatonka +	112	6.89	chytili a odvlekli			boilies	10	5.78
chytil zlatonku			popravit	45	5.69	třpytka	7	5.54
slina +	178	6.80	chycen a popraven			flop	13	5.48
chytil slinu			dohonit	24	5.31	chytil na flopu		
rybka +	200	6.72	dohonit a chytit			boilie	8	5.40
chytí zlatou rybku			okroužkovat	14	5.31	háček	53	5.36
štika +	115	6.51	chytit a okroužkovat			švestka	21	5.18
chytil štiku			odtáhnout	38	5.21	chytit na švestkách		

Fig. 5: Word sketches for the word *chytit*

As for word sketches (Figure 5), the *coord* category has only one potentially applicable item with high score, antonym *pustit* (to drop, to let go). Other items (thief, breath, etc.) can be interesting but it is not clear how to use them directly.

4.3 Verbs: Summary

It is obvious that verbs require a different approach than nouns. Current corpus tools do not in fact offer any efficient way how to create definitions similar to those used in SSČ; however the word sketch results show objects that could help to describe meaning.

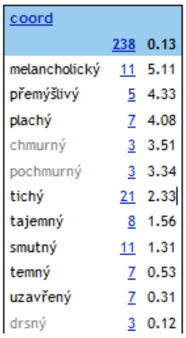
5 Adjectives

5.1 Zádumčivý (broody)

According to SSČ, *zádumčivý* has two meanings:

Table 6: Thesaurus results for *zádumčivý*

Lemma	Translation	Score	Freq
zadumaný	pensive	0.369	1,924
zasmušilý	melancholic	0.257	1,982
tklivý	touching	0.235	2,441
snivý	dreamful	0.233	1,432
zamyšlený	wistful	0.227	5,183
posmutnělý	unhappy	0.227	2,504
zasněný	wistful	0.215	5,095
teskný	sorrowful	0.205	2,074
introvertní	introvert	0.189	3,388
zamlklý	taciturn	0.181	5,388



coord		
	238	0.13
melancholický	11	5.11
přemýšlivý	5	4.33
plachý	7	4.08
chmurný	3	3.51
pochmurný	3	3.34
tichý	21	2.33
tajemný	8	1.56
smutný	11	1.31
temný	7	0.53
uzavřený	7	0.31
drsný	3	0.12

Fig. 6: Word sketches for the word *zádumčivý*

1. zasmušilý (melancholic)
2. působcí takovým dojmem, smutný (looking broody; sad)

Table 6 shows that it is most similar to words *zadumaný* (pensive) and *zasmušilý* (melancholic). Word sketch relation *coord* (see Figure 6) displays quite similar results. Basically, all the results present there are synonyms.

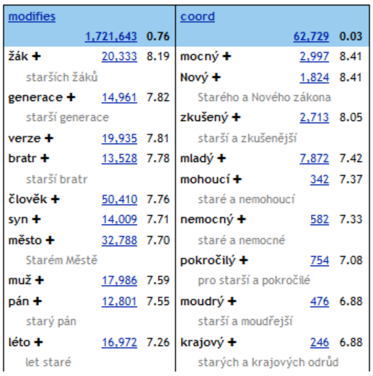
5.2 Starý (old)

SSČ provides eleven meanings of *starý*:

1. jsoucí v závěrečném období života, vysokého věku (nearing the end of life, of advanced age)
2. (o člověku) jsoucí urč. věku, vytvořený před urč. dobou (being of certain age, created certain time ago)
3. v stáří obvyklý, stáří vlastní (typical to old age)
4. vytvořený před delší dobou, dlouhým užíváním opotřebovaný, bezcenný (created long time ago, timeworn, obsolete)
5. jsoucí dávného původu (being old, ancient)

Table 7: Thesaurus results for *starý*

Lemma	Translation	Score	Freq
nový	new	0.494	6,374,792
známý	known	0.433	1,412,791
původní	original	0.415	839,282
samotný	alone	0.407	932,916
velký	big	0.406	7,849,239
mladý	young	0.404	1,632,302
vlastní	one's own	0.399	1,877,789
krásný	beautiful	0.398	1,301,992
jediný	only	0.394	1,678,394
kvalitní	superior	0.391	941,081



modifies		
	1,721,643	0.76
žák +	20,333	8.19
staršich žáků		
generace +	14,961	7.82
starší generace		
verze +	19,935	7.81
bratr +	13,528	7.78
starší bratr		
člověk +	50,410	7.76
syn +	14,009	7.71
město +	32,788	7.70
Starém Městě		
muž +	17,986	7.59
pán +	12,801	7.55
starý pán		
léto +	16,972	7.26
let staré		

coord		
	62,729	0.03
mocný +	2,997	8.41
Nový +	1,824	8.41
Starého a Nového zákona		
zkušený +	2,713	8.05
starší a zkušenější		
mladý +	7,872	7.42
mohoucí +	342	7.37
staré a nemohoucí		
nemocný +	582	7.33
staré a nemocné		
pokročilý +	754	7.08
pro starší a pokročilé		
moudrý +	476	6.88
starší a moudřejší		
krajový +	246	6.88
starých a krajových odrůd		

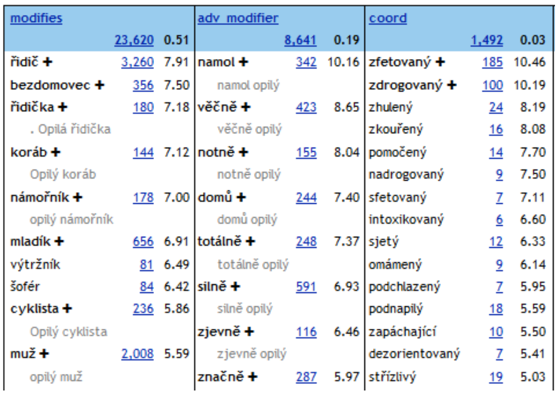
Fig. 7: Word sketches for the word *starý*

6. zastaralý, nemoderní (archaic, outdated)
7. dávno známý, často opakovaný (known for a long time, often repeated)
8. předešlý, bývalý (former)
9. dávný (ancient)
10. stejný jako dříve, původní (same as before, original)
11. delší dobu něco konající, osvědčený, zkušený (doing sth for a long time, time-proven, experienced)

Thesaurus places the word *nový* in the first place (Table 7); it is an antonym (as well as *mladý*). The only synonymic meaning in the first ten results is *původní* (original) which does not describe the primary meaning of *starý*. The word sketch data are not of much use either, as shown in Figure 7; it shows who and what can be old (in column *modifies*) which is almost anything. Antonyms *nový* and *mladý* can be useful which are also in thesaurus.

Table 8: Thesaurus results for *opilý*

Lemma	Translation	Score	Freq
ožralý	drunk	0.404	6,877
místný	local	0.321	87,162
sympatický	likable	0.314	107,989
podnapilý	tipsy	0.304	10,123
naštvaný	angry	0.303	52,656
sedící	sitting	0.302	34,073
vyděšený	scared	0.298	24,121
letý	of age	0.297	184,909
unavený	tired	0.289	101,496
nebohý	pathetic	0.282	17,633



modifies			adv_modifier			coord		
	23,620	0.51		8,641	0.19		1,492	0.03
řidič +	3,260	7.91	namol +	342	10.16	zfetovaný +	185	10.46
bezdomovec +	356	7.50	namol opilý			zdrogovaný +	100	10.19
řidička +	180	7.18	věčně +	423	8.65	zhulený	24	8.19
. Opilá řidička			věčně opilý			zkouřený	16	8.08
koráb +	144	7.12	notně +	155	8.04	pomočený	14	7.70
Opilý koráb			notně opilý			nadrogovaný	9	7.50
námořník +	178	7.00	domů +	244	7.40	sfetovaný	7	7.11
opilý námořník			domů opilý			intoxikovaný	6	6.60
mladík +	656	6.91	totálně +	248	7.37	sjetý	12	6.33
výtržník	81	6.49	totálně opilý			omámený	9	6.14
šofér	84	6.42	silně +	591	6.93	podchlazený	7	5.95
cyklista +	236	5.86	silně opilý			podnapilý	18	5.59
Opilý cyklista			zjevně +	116	6.46	zapáchající	10	5.50
muž +	2,008	5.59	zjevně opilý			dezorientovaný	7	5.41
opilý muž			značně +	287	5.97	střízlivý	19	5.03

Fig. 8: Word sketches for the word *opilý*

5.3 Opilý (drunk)

Opilý has three meanings defined in SSČ:

1. opojený nadměrným požitím alkoholického nápoje (intoxicated by alcohol)
2. svědčící o tom (indicating sb is drunk)
3. expr. mocně zaujatý, opojený, omámený (really preoccupied, intoxicated)

Table 8 shows that the most similar thesaurus result is the expressive synonym *ožralý*. Word sketch does not show any applicable results, maybe except for *namol (opilý)* (blind drunk) from the *adv_modifier* category, and semantically near words in the *coord* category.

5.4 Adjectives: Summary

While it is possible to define some adjectives using existing corpus tools, in other cases it seems to be more complicated. The difference might be in how many meanings a word can have and how narrow these meanings are.

6 Synsemantics

6.1 Poněvadž (because)

Poněvadž is defined as

> sp. podř. příčin. (důvod.), protože (subordinating conjunction expressing a cause)

Thesaurus as well as word sketch offers numerous synonyms (see Table 9 and Figure 9).

6.2 Nebo (or)

Nebo is defined as

> 1. vyj. vztah neslučitelnosti, anebo (expression of contradictoriness)

or

> 2. vyj. vztah mezi dvěma i více možnostmi, i časovými (relation between two and more options)

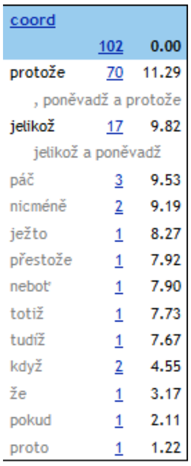
Here the data are not so clear, thesaurus suggests synonym *či* (Table 10), however, as shown in Figure 10, word sketch only provides few results. Relation *post_inf* may be interesting to look at: if we had also relation *prec_inf*, we may be able to extract useful usage patterns, such as *potvrdit nebo vyvrátit* (to confirm or disprove).

6.3 Synsemantics: Summary

As long as conjunctions are defined only by synonyms, it is possible to obtain some useful results; however, this would inevitably lead to a circular definition. The question is how the definitions of synsemantics should look like. The SSČ-like definitions would be hard to create, but they do not seem to be extremely useful anyway. Usage patterns may be a better way of "defining" these words, and it may be easier to extract them from corpora.

Table 9: Thesaurus results for *poněvadž*

Lemma	Translation	Score	Freq
anžto	because	0.588	2,565
páč	because	0.373	77,458
jelikož	because	0.346	539,891
jenomže	only	0.291	85,636
přestože	although	0.288	391,756
jestliže	if	0.250	363,826
bo	because	0.225	48,891
byť	although	0.218	247,035
neboť	because	0.156	767,296
nýbrž	but	0.148	206,053



coord		
	102	0.00
protože	70	11.29
, poněvadž a protože		
jelikož	17	9.82
jelikož a poněvadž		
páč	3	9.53
nicméně	2	9.19
ježto	1	8.27
přestože	1	7.92
neboť	1	7.90
totiž	1	7.73
tudíž	1	7.67
když	2	4.55
že	1	3.17
pokud	1	2.11
proto	1	1.22

Fig. 9: Word sketches for the word *poněvadž*

7 Conclusions

As shown above, the existing corpus tools are able to find fragments that can be used in definitions. Different parts of speech will require slightly different approaches.

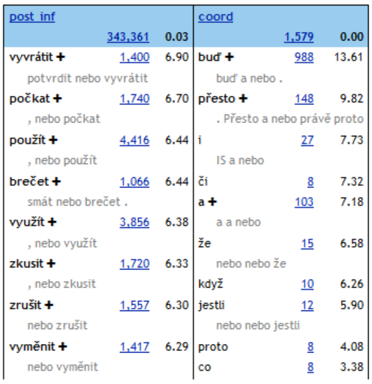
It is impossible to estimate whether automatic definition of synsemantics will prove doable, because their meaning is often too specific to be defined easily (e. g. conjunctions (and, or) and prepositions (in, on)).

Automatic creation of definitions, at least to some extent, should be possible for nouns, verbs, adjectives and adverbs. A special sketch grammar aimed at needs of such definitions may help.

Existing dictionaries (like SSČ) list many meanings which are quite similar and listing all those meanings is redundant. Differentiation between meanings that should be distinguished is another task.

Table 10: Thesaurus results for *nebo*

Lemma	Translation	Score	Freq
či	or	0.908	3,395,720
,		0.884	303,403,489
a	and	0.879	137,863,596
i	also	0.869	25,577,373
-		0.850	18,915,328
)		0.840	28,670,317
on	he	0.834	32,844,994
:		0.817	27,939,280
ale	but	0.816	25,330,812
ani	neither	0.803	5,674,148



post inf			coord		
	343,361	0.03		1,579	0.00
vyvrátit +	1,400	6.90	buď +	988	13.61
potvrdit nebo vyvrátit			buď a nebo .		
počkat +	1,740	6.70	přesto +	148	9.82
, nebo počkat			. Přesto a nebo právě proto		
použít +	4,416	6.44	i	27	7.73
, nebo použít			IS a nebo		
brečet +	1,066	6.44	či	8	7.32
smát nebo brečet .			a +	103	7.18
využít +	3,856	6.38	a a nebo		
, nebo využít			že	15	6.58
zkusit +	1,720	6.33	nebo nebo že		
, nebo zkusit			když	10	6.26
zrušit +	1,557	6.30	jestli	12	5.90
nebo zrušit			nebo nebo jestli		
vyměnit +	1,417	6.29	proto	8	4.08
nebo vyměnit			co	8	3.38

Fig. 10: Word sketches for the word *nebo*

Acknowledgments. This work has been partly supported by the Masaryk University within the project *Čeština v jednotě synchronie a diachronie – 2016* (MUNI/A/0863/2015) and by the Ministry of Education of CR within the LINDAT-Clarin project LM2015071.

References

1. HarperCollins Publishers: The history of COBUILD. [online] ((accesed November 9, 2016))
2. Kovář, V., Močiariková, M., Rychlý, P.: Finding definitions in large corpora with Sketch Engine. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016), Paris, France, European Language Resources Association (ELRA) (2016)
3. Przepiórkowski, A., Degórski, Ł., Wójtowicz, B., Spousta, M., Kuboň, V., Simov, K., Osenova, P., Lemnitzer, L.: Towards the automatic extraction of definitions in Slavic. In: Proceedings of the Workshop on Balto-Slavonic Natural Language Processing: Information Extraction and Enabling Technologies, Association for Computational Linguistics (2007) 43–50

4. Navigli, R., Velardi, P., Ruiz-Martínez, J.M.: An annotated dataset for extracting definitions and hypernyms from the web. In: Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10), Valletta, Malta, European Language Resources Association (ELRA) (2010)
5. Jin, Y., Kan, M.Y., Ng, J.P., He, X.: Mining scientific terms and their definitions: A study of the ACL anthology. EMNLP-2013 (2013)
6. Klavans, J.L., Muresan, S.: Evaluation of the DEFINDER system for fully automatic glossary construction. In: Proceedings of the AMIA Symposium, American Medical Informatics Association (2001) 324
7. Čermák, F., Blatná, R.: Manuál lexikografie. 1st edn. H & H, Jinočany (1995)
8. Filipec, J.: Slovník spisovné češtiny pro školu a veřejnost. 4th edn. Academia, Praha (2005)
9. Kilgarriff, A.: "I don't believe in word senses". Computers and the Humanities **31**(2) (1997) 91–113

Part IV

Corpora-based Applications

Evaluating Natural Language Processing Tasks with Low Inter-Annotator Agreement: The Case of Corpus Applications

Vojtěch Kovář

Natural Language Processing Centre
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
xkovar3@fi.muni.cz

Abstract. In [1], we have argued that tasks with low inter-annotator agreement are really common in natural language processing (NLP) and they deserve an appropriate attention. We have also outlined a preliminary solution for their evaluation. In [2], we have agitated for extrinsic application-based evaluation of NLP tasks and against the gold standard methodology which is currently almost the only one really used in the NLP field.
This paper brings a synthesis of these two: For three practical tasks, that normally have so low inter-annotator agreement that they are considered almost irrelevant to any scentific evaluation, we introduce an application-based evaluation scenario which illustrates that it is not only possible to evaluate them in a scientific way, but that this type of evaluation is much more telling than the gold standard way.

Key words: NLP, inter-annotator agreement, low inter-annotator agreement, evaluation, application, application-based evaluation, word sketch, thesaurus, terminology

1 Introduction

1.1 Gold standard evaluation methodology

Scientific evaluation of applications in the natural language processing (NLP) field is usually based on so-called *gold standards* – data sets that contain "correct" annotations created mostly by human beings who understand the particular language (and often also the the underlying linguistic theory). In this type of evaluation, we measure the similarity between this gold standard and an output of a particular tool that is being tested.

For example, in case of morphological analysis, such a gold standard is a corpus manually annotated with morphological tags. In case of syntactic analysis, it is a treebank (corpus where each sentence is manually annotated with a syntactic tree). For machine translation, it is a corpus of correct translations. Similarity metrics for these cases usually are:

- percentage of morphological tags that are identical in both gold standard and on the output of a tagger
- various types of tree similarity metrics [3,4,5]
- the famous BLEU score [6] and its modifications

1.2 Problems with gold standards

This methodology, however, has significant drawbacks. In [2], we argued that it often does not measure the important bits of the linguistic information; that the NLP tools often overfit to the gold standards and therefore their output is often not suitable for practical applications; that there is almost no ambiguity allowed in a typical gold standard; or that the particular evaluation results crucially depend on arbitrary decisions taken at the time of building the gold standard.

As we explain in [2], inter-annotator agreement (IAA) is another issue; it is one of the most important and most problematic aspects of gold standard evaluations. Although high IAA is usually considered crucial for the task to be "well-defined", it is rarely officially published. Often, the lack of agreeent is addressed by extensive annotation manuals (one example for all: annotation guide to a tectogrammatical layer of syntactic annotation in the Prague Dependency Treebank [7] with more than 1200 pages!) that are impossible to memorize – which (apart from frequent errors and inconsistencies) leads to the annotations being record of all the arbitrary decisions present in the manual, rather than native speaker language intuition.

However, there is one problem that is even more important: For some tasks, such as collocation extraction, building an automatic thesaurus, or terminology extraction, the IAA is so low that it is almost impossible to build gold standards for them [8,9], and thus they are doomed to be considered ill-defined and not suitable for scientific evaluation. However, these applications are far from being useless, rather the opposite: there are quite strong commercial interests in them, as can be illustrated e.g. by the successful Sketch Engine service [10] – and we need to be able to evaluate them in a scientific way!

1.3 What this paper is about

In [1], we argued that applications with low IAA should not be considered inferior and that we should find a way to evaluate them. We also introduced a preliminary evaluation methodology for these low-IAA tasks, still based on the gold standard methodology. This paper presents a shift from the gold standards to the purely application-based methodology, and presents a concrete evaluation methods for the three already mentioned applications: collocation extraction, as in the word sketches [10], automatic (distributional) thesaurus generation, and terminology extraction.

All of these are commercially interesting applications that are around already for a rather long time, but so far have not been sufficiently evaluated. The idea we present here is basically very simple: for the current users, the output of these applications output is useful as it is – so it is the output itself that should be evaluated, and it must be the users who evaluate it (rather than a combination of a gold standard and a similarity measure).

2 Evaluation methodology

The proposed methodology follows the general idea presented in [2]: We present two different versions of the particular application output to the group of users/evaluators; we highlight differences, and the users (evaluators) will decide which parts of which version are better/worse (and, possibly, how much better/worse).

Then we sum the overall results from all the evaluators – this will give us one number that expresses which version is better. Note that it does not matter if the annotators agree with each other or not; the solution with more votes is winning, no matter how different the evaluator's opinions are (this may seem unfair but it simulates the real-world situation).

This evaluation methodology allows a lot of options in number of evaluators, the exact evaluation method, testing sample etc. – all of these aspects will influence the quality and the soundness of the evaluation. On the other hand, this variability also enables evaluation of the applications in different usage scenarios.

Also, as we discussed in [2], this method has its drawbacks – it may be more expensive, less sensitive, more suitable for cheating etc. – but its main feature is priceless: The application is evaluated by real users in real usage scenarios, and it is directly the application that is being evaluated, not an artificial "middleware" which may or may not be important (such as syntactic analysis according to a particular treebank annotation).

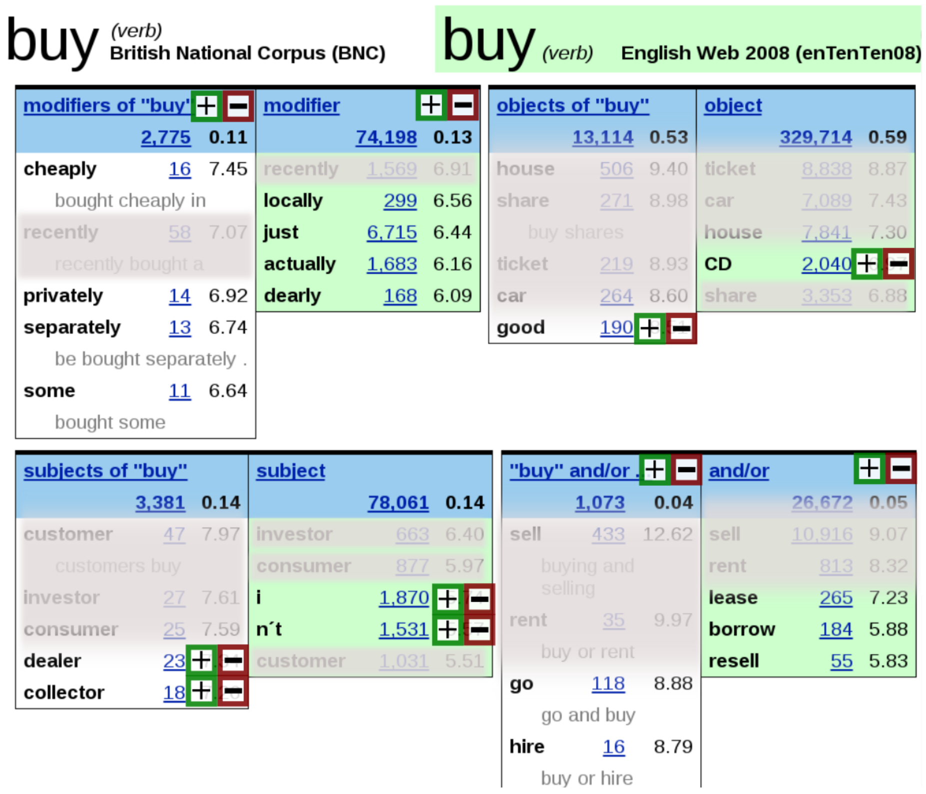
In the following sections, we propose the particular evaluation set-ups for the three already mentioned applications.

3 Collocation extraction

Word sketch [10] is the state of the art application for collocation extraction from corpora. Therefore, we take it as the base for our evaluation. The evaluation will compare two different settings of the word sketch application.

We propose the following evaluation setup:

- we select a set of sample words (may represent a general language use, or can be more specialized)
- for each of the sample words, we display two word sketches on one page, particular relations aligned to each other
- when the relations are very different, we put +/- buttons to the relations
- when the relations only differ in 1 or 2 (maybe 3) words, we put +/- buttons to the particular words
- we hide the common parts



buy (verb) British National Corpus (BNC)

modifiers of "buy" ＋ −	2,775	0.11
cheaply	16	7.45
bought cheaply in		
recently	58	7.07
recently bought a		
privately	14	6.92
separately	13	6.74
be bought separately .		
some	11	6.64
bought some		

objects of "buy"	13,114	0.53
house	506	9.40
share	271	8.98
buy shares		
ticket	219	8.93
car	264	8.60
good	190	＋ −

subjects of "buy"	3,381	0.14
customer	47	7.97
customers buy		
investor	27	7.61
consumer	25	7.59
dealer	23	＋ −
collector	18	＋ −

"buy" and/or . ＋ −	1,073	0.04
sell	433	12.62
buying and selling		
rent	35	9.97
buy or rent		
go	118	8.88
go and buy		
hire	16	8.79
buy or hire		

buy (verb) English Web 2008 (enTenTen08)

modifier ＋ −	74,198	0.13
recently	1,569	6.91
locally	299	6.56
just	6,715	6.44
actually	1,683	6.16
dearly	168	6.09

object	329,714	0.59
ticket	8,838	8.87
car	7,089	7.43
house	7,841	7.30
CD	2,040	＋ −
share	3,353	6.88

subject	78,061	0.14
investor	663	6.40
consumer	877	5.97
i	1,870	＋ −
n´t	1,531	＋ −
customer	1,031	5.51

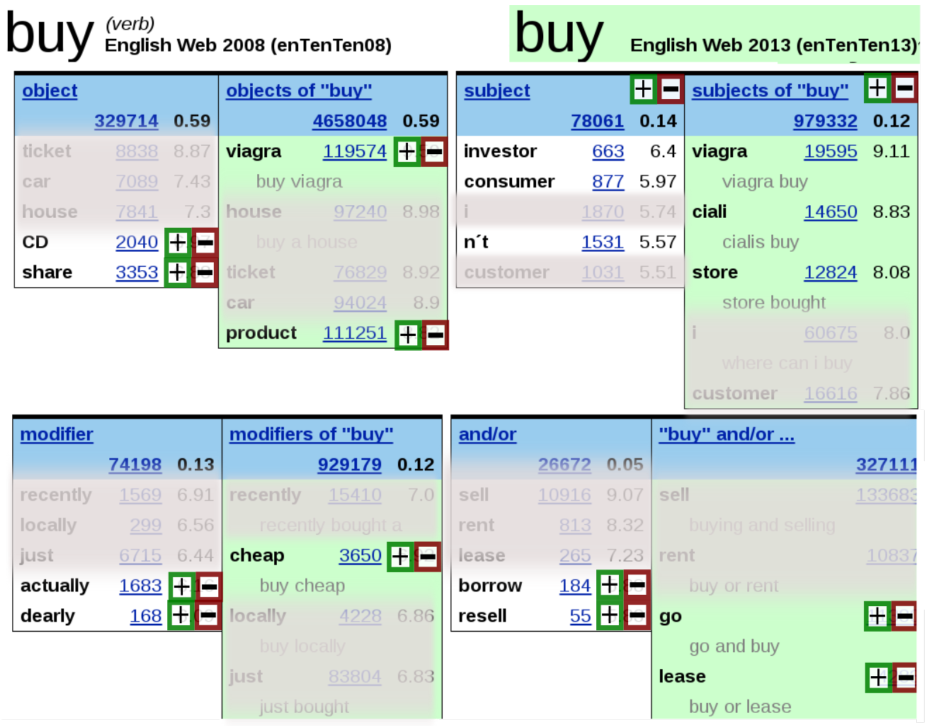
and/or ＋ −	26,672	0.05
sell	10,916	9.07
rent	813	8.32
lease	265	7.23
borrow	184	5.88
resell	55	5.83

Fig. 1: Word sketch evaluation proposal: British National Corpus vs. English web corpus enTenTen08

- each evaluator can click each button several times (to express different importance of the differences) but they are not obliged to click anything (to be able to express that something is not really important)
- at the end of the evaluation, we count +1/-1 point for every +/- click on a collocate, +2/-2 points for every click on a relation; the overall sum is the result

Figures 1 and 2 illustrate the particular examples of what the evaluators would see – in some cases, it is perfectly clear which side is better (e.g. the "n't" collocate is a result of a processing error, "viagra" etc. is a result of web spam present in the corpus), in other cases the opinions may differ.

The figures contain the names of the two corpora but this is only for illustration purposes. In reality we would not show the different settings to the evaluators; firstly because their opinions could be biased by the corpus names, and also because we can measure a wide range of different settings (e.g. different minimum frequency), not just a different corpus.



buy (verb) English Web 2008 (enTenTen08)

object		
	329714	0.59
ticket	8838	8.87
car	7089	7.43
house	7841	7.3
CD	2040	
share	3353	

objects of "buy"		
	4658048	0.59
viagra	119574	
buy viagra		
house	97240	8.98
buy a house		
ticket	76829	8.92
car	94024	8.9
product	111251	

modifier		
	74198	0.13
recently	1569	6.91
locally	299	6.56
just	6715	6.44
actually	1683	
dearly	168	

modifiers of "buy"		
	929179	0.12
recently	15410	7.0
recently bought a		
cheap	3650	
buy cheap		
locally	4228	6.86
buy locally		
just	83804	6.83
just bought		

buy English Web 2013 (enTenTen13)

subject		
	78061	0.14
investor	663	6.4
consumer	877	5.97
i	1870	5.74
n´t	1531	5.57
customer	1031	5.51

subjects of "buy"		
	979332	0.12
viagra	19595	9.11
viagra buy		
ciali	14650	8.83
cialis buy		
store	12824	8.08
store bought		
i	60675	8.0
where can i buy		
customer	16616	7.86

and/or		
	26672	0.05
sell	10916	9.07
rent	813	8.32
lease	265	7.23
borrow	184	
resell	55	

"buy" and/or ...	
	327111
sell	133683
buying and selling	
rent	10837
buy or rent	
go	
go and buy	
lease	
buy or lease	

Fig. 2: Word sketch evaluation proposal: Older English web corpus enTenTen08 vs. newer English web corpus enTenTen13

4 Thesaurus

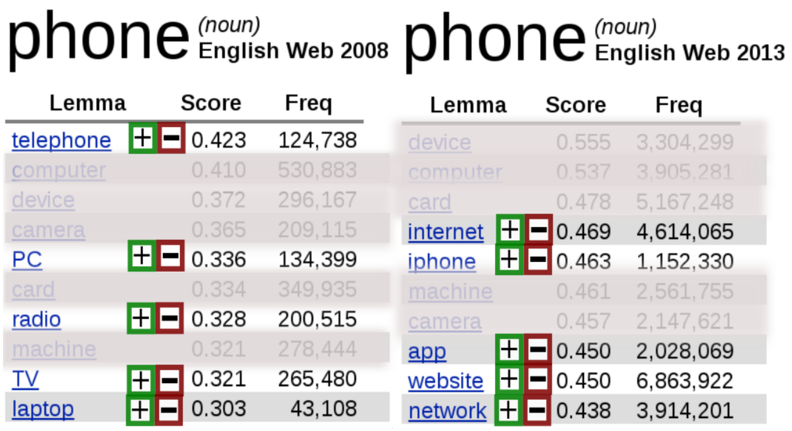
Thesaurus is basically just a list of similar words, so the task reduces to comparison of two lists, again for a given sample of words.

The scenario here is very similar to what we propose in case of collocation extraction: take the top of both lists, put the two lists side by side, ignore common items and evaluate individual items on the list by clicking +/-. Sum of positive and negative points is then the score of a particular list.

An example of what the annotators would see is in Figure 3. Again, the corpus names would not be shown.

5 Terminology extraction

Extraction of terminology from domain-specific texts is in fact another list, so the procedure can be very similar to the thesaurus evaluation, as introduced above. There is just one difference: the terminology is not for one particular word, but for a whole corpus, so no sample of words is needed here. Rather



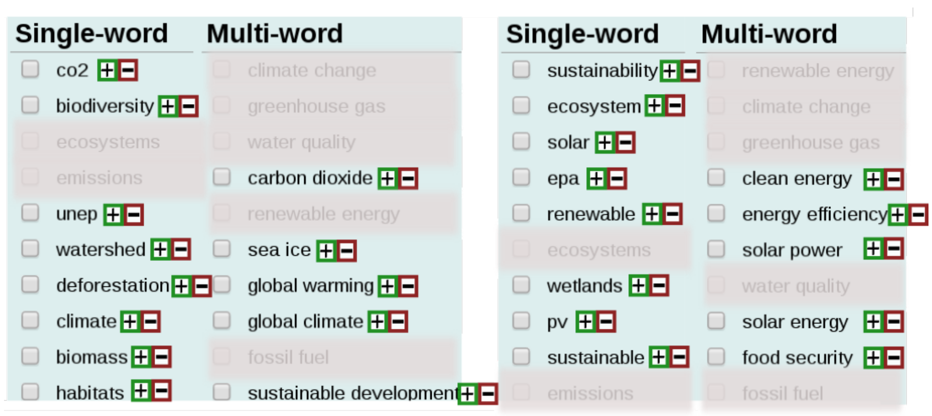
phone *(noun)* **English Web 2008**

Lemma	Score	Freq
telephone	0.423	124,738
computer	0.410	530,883
device	0.372	296,167
camera	0.365	209,115
PC	0.336	134,399
card	0.334	349,935
radio	0.328	200,515
machine	0.321	278,444
TV	0.321	265,480
laptop	0.303	43,108

phone *(noun)* **English Web 2013**

Lemma	Score	Freq
device	0.555	3,304,299
computer	0.537	3,905,281
card	0.478	5,167,248
internet	0.469	4,614,065
iphone	0.463	1,152,330
machine	0.461	2,561,755
camera	0.457	2,147,621
app	0.450	2,028,069
website	0.450	6,863,922
network	0.438	3,914,201

Fig. 3: Thesaurus evaluation proposal: Older English web corpus enTenTen08 vs. newer English web corpus enTenTen13



Single-word	Multi-word
co2	climate change
biodiversity	greenhouse gas
ecosystems	water quality
emissions	carbon dioxide
unep	renewable energy
watershed	sea ice
deforestation	global warming
climate	global climate
biomass	fossil fuel
habitats	sustainable development

Single-word	Multi-word
sustainability	renewable energy
ecosystem	climate change
solar	greenhouse gas
epa	clean energy
renewable	energy efficiency
ecosystems	solar power
wetlands	water quality
pv	solar energy
sustainable	food security
emissions	fossil fuel

Fig. 4: Terminology evaluation proposal: 60M Environment domain corpus with two different reference corpora: big (11 billion words) web corpus enTenTen12 on the left, and small (7 million words) manually created corpus Brown Family.

than that, we need a sample of domain specific texts. Again, the results may be very different for different samples, however, this reflects the reality: A terminology extractor can also be very good on one domain and very bad on another one.

The fact that no sample of words is needed means that we can include more items into the list, not just 10 or 20 as in case of word sketches and thesaurus. And we really should do that because terminology extraction has a different use case than the two other applications. In both word sketch and thesaurus, the user typically looks at up to 20 top items; in case of terminology, thousands of items may be extracted (e.g. for the purpose of compiling a specialized dictionary) to be further processed.

We should always be sure that we are testing something that is as close to the real use case, to the real application of the particular tool, as possible.

An example of what the annotators would see is in Figure 4 – but as we've just explained, in reality the lists would be longer (and because of that, they would probably also contain more hidden items).

6 Conclusions

Based on previous theoretical work, we have introduced a concrete scenario of application-based evaluation of three NLP tasks with low inter-annotator agreement. We believe this proposal will be implemented in a short time and used as an evaluation framework for these tasks.

Future work consists mainly in actually *doing* a robust evaluation of these three tasks according to the scenarios introduced in this paper, for various corpora and various settings.

Acknowledgments. The research leading to these results has received funding from the Norwegian Financial Mechanism 2009–2014 and the Ministry of Education, Youth and Sports under Project Contract no. MSMT-28477/2014 within the HaBiT Project 7F14047.

References

1. Kovář, V., Rychlý, P., Jakubíček, M.: Low inter-annotator agreement = an ill-defined problem? In: Eighth Workshop on Recent Advances in Slavonic Natural Language Processing, Brno, Tribun EU (2014) 57–62
2. Kovář, V., Jakubíček, M., Horák, A.: On evaluation of natural language processing tasks: Is gold standard evaluation methodology a good solution? In: Proceedings of the 8th International Conference on Agents and Artificial Intelligence, Rome, SCITEPRESS (2016) 540–545
3. Grishman, R., Macleod, C., Sterling, J.: Evaluating parsing strategies using standardized parse files. In: Proceedings of the third conference on Applied natural language processing, Association for Computational Linguistics (1992) 156–161
4. Sampson, G.: A proposal for improving the measurement of parse accuracy. International Journal of Corpus Linguistics **5**(01) (2000) 53–68
5. Sampson, G., Babarczy, A.: A test of the leaf-ancestor metric for parse accuracy. Natural Language Engineering **9**(04) (2003) 365–380
6. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: BLEU: a method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting on association for computational linguistics, Association for Computational Linguistics (2002) 311–318

7. Mikulová, M., Bémová, A., Hajič, J., Hajičová, E., Havelka, J., Kolářová, V., Kučová, L., Lopatková, M., Pajas, P., Panevová, J., Razímová, M., Sgall, P., Štěpánek, J., Urešová, Z., Veselá, K., Žabokrtský, Z.: Annotation on the tectogrammatical layer in the Prague Dependency Treebank (2005) `http://ufal.mff.cuni.cz/pdt2.0/doc/manuals/en/t-layer/pdf/t-man-en.pdf`.
8. Kilgarriff, A., Kovář, V., Krek, S., Srdanović, I., Tiberius, C.: A quantitative evaluation of Word Sketches. In: Proceedings of the XIV Euralex International Congress, Ljouwert, Netherlands, Fryske Akademy (2010) 372–379
9. Kilgarriff, A., Rychlý, P., Jakubíček, M., Kovář, V., Baisa, V., Kocincová, L.: Extrinsic corpus evaluation with a collocation dictionary task. In: Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14), Reykjavik, Iceland, European Language Resources Association (ELRA) (2014) 1–8
10. 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)

Terminology Extraction for Academic Slovene Using Sketch Engine

Darja Fišer[1,2], Vít Suchomel[3,4], Miloš Jakubíček[3,4]

[1] Dept. of Translation, Faculty of Arts, University of Ljubljana,
Aškerčeva cesta 2, SI-1000 Ljubljana, Slovenia
[2] Department of Knowledge Technologies, Jožef Stefan Institute,
Jamova cesta 3, SI-1000 Ljubljana, Slovenia
darja.fiser@ff.uni-lj.si

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

[4] Lexical Computing
Brighton, United Kingdom and Brno, Czech Republic
{vit.suchomel,milos.jakubicek}@sketchengine.co.uk

Abstract. In this paper we present the development of the terminology extraction module for Slovene which was framed within the Sketch Engine corpus management system and motivated by the KAS research project on resources and tools for analysing academic Slovene. We describe the formalism used for defining the grammaticality of terms as well as the calculation of the score of individual terms, give an overview of the definition of the term grammar for Slovene and evaluate it on a Slovene KAS corpus of academic Slovene.

Key words: terminology, term extraction, Sketch Engine, academic discourse, Slovene

1 Introduction

The development of the academic part of any language is an important indicator of its vitality [1,2,3]. A key component of any scientific communication is terminology which needs to be analysed by linguistcs and terminologists but also has to be made easily accessible to domain experts, such as researchers, lecturers and practicioners, as well as to students, translators and editors [4,5]. Since authoritative terminological dictionaries are always lagging behind the fast-paced development of terminology in the scientific domain, corpus-driven and collaborative approaches to terminology management have become an attractive alternative in the past decades [6,7], also for Slovene [8].

To achieve this for Slovene, we have compiled a large KAS corpus of 50,000 scientific texts with over one billion tokens [9] and are now in the process of developing term extraction for it, which is the subject of this paper.

The paper is organised as follows: in the next section we present an adaptation of the terminology extraction module in Sketch Engine, a leading corpus management tool [10] and its terminology extraction methodology. Next, we outline the term grammar for Slovene and finally conclude with an evaluation of the terms extracted from the KAS corpus of academic Slovene that will provide useful insight for future refinement of the term extraction tool and the term grammar.

2 The Sketch Engine Environment

Sketch Engine is an online corpus management system providing access to hundreds of text corpora which can be searched and analyzed. It received its name after one of its key features—word sketches, one page summaries of a word's collocational behaviour in particular grammatical relations.

As of 2016, Sketch Engine hosts preloaded corpora for 85 languages and allows users to create new ones, either by uploading their own texts or by building the corpus semi-automatically from the web according to the keywords given by the user. The latter approach is particularly useful for fast development of domain corpora from online resources.

2.1 The Terminology Extraction Module

Sketch Engine contains a terminology extraction module [11] using a contrastive approach for finding term candidates, in a similar manner to other such systems [12,13]. Two corpora are given as input to the term extraction: a focus corpus consisting from texts in the target domain, and a (ideally very big) reference corpus against which the focus corpus is then compared. To improve the accuracy of this process, Sketch Engine selects only grammatically valid phrases.

Therefore the whole extraction is a two-step process:

1. **unithood**: the first step is rule based and language dependent. We assess the grammatical validity of a phrase (*unit hood*) using a so called *term grammar*. A term grammar describes grammatically plausible terms using regular expressions over available annotation in the corpus, such as morphosyntactic tags and lemmas.
2. **termhood**: candidate phrases generated in the first step are then contrasted with the reference corpus by using the "simplemath" statistic [14] which compares their normalized frequencies focusing either on less frequent or more frequent phrases.

The result of the term extraction is a list of (multi-word) term candidates sorted according to their simplemath score.

2.2 Term Grammar for Slovene

The Slovene term grammar v1.0 is based on the original term definition for Russian by Maria Khokhlova and the transformed term definition for Czech by Vít Suchomel. From these sources it has then been considerably extended with additional term patterns needed for a comprehensive terminological analysis of the KAS corpus of academic Slovene.

Default Attrubutes. The grammar defines 12 default attributes, 10 of which are MSD-based to ensure high-accuracy term candidate identification and 2 serve to achieve agreement in gender, number and/or case across the multi-word term for improved accuracy of term identification.

Term Patterns. The main part of the term grammar are term patterns that use combinations of the defined default attributes to identify and render the extracted term candidates. The following parts of speech were considered as possible elements of term patterns: noun, adjective, preposition, conjunction, adverb and verb. v1.0 of the Slovene term grammar enables term extraction of single as well as multi-word terms consisting of noun as well as verb phrases up to length 4.

In total, 44 term patterns have been defined:

- 4-grams: 22 patterns
- 3-grams: 15 patterns
- bigrams: 6 patterns
- unigrams: 1 pattern

An illustrative example of a term pattern is given in Figure 1. The first line contains rendering instructions for the given pattern. According to it, the first word should be rendered as a lowercased lemma while the rest of the elements should be displayed as lowercased word forms. The second line contains the pattern to be identified in the corpus. The rule will identify all the 4-grams in the corpus that start with a noun and are followed by 2 consecutive adjectives and a noun in the genitive case. In addition, the pattern requires that the second and the third element in the term agree with the final noun in gender, number and case. The third line contains the CQL version of the same pattern suitable for corpus querying to facilitate development and editing of the term grammar. With a similar purpose, the last line shows an example of such a pattern from the corpus.

3 Evaluation

3.1 KAS-PhD Corpus

An evaluation of the term extraction module and the term grammar for Slovene was performed on the KAS subcorpus of 700 PhD theses which contains almost

```
*COLLOC "%(1.lemma_lc)_%(2.lc)_%(3.lc)_%(4.lc)-x"
1:noun 2:adj_genitive 3:adj_genitive 4:noun_genitive & agree(2,4) & agree(3,4)
#"Nc.*" "A.*g.*" "A.*g.*" "Nc.*g.*"
#metoda magnetronskega ionskega naprševanja
```

Fig. 1: Example of a term pattern in term grammar.

150,000 pages of text or 53 million tokens published in the period 2000-2015. Most theses in the corpus are from Social and Technical Sciences, some are from Natural Sciences while there are very few from Biomedical Sciences and the Humanities. While terminology is typically extracted for a limited domain, our main goal in this paper was to evaluate the term grammar, which is why we believe using a heterogeneous corpus is more suitable as it will highlight different characteristics and issues across several domains.

3.2 Results

The evaluation was performed on the 1,000 top-ranking 3- and 4-gram term candidates from the KAS-PhD corpus with respect to the reference slTenTen corpus of general Slovene [15]. A large majority of them were bigrams, with only a few 3- and 4-grams:

- 4-grams: 28 (2.8%) term candidates
- 3-grams: 177 (17.7%) term candidates
- bigrams: 795 (79.5%) term candidates

Manual evaluation consisted of three steps. First, pattern productivity was considered in order to determine which patterns in the term grammar have a good yield. Next, term candidates were checked for unithood and structural accuracy so as to identify any remaining bugs in the term grammar. In the end, termhood of the extracted candidates was tested, the goal of which was to suggest further refinements of term ranking and smoothing.

Results for 4-grams. As there were only 28 4-gram candidates, all were manually examined. By far the most productive patterns in this category of the extracted term candidates are noun phrases that contain a preposition (68%, e.g. `družba z omejeno odgovornostjo`, followed by the much less productive combinations of adjectives and nouns (18%, e.g. `zaznana vrednost blagovne znamke`) and patterns that contain a conjunction (14%, e.g. `mera središčnosti in pomembnosti`).

In terms of structural accuracy and unithood, patterns containing prepositions or conjunctions significantly outperform adjective+noun combinations (75% wrt. 40%), indicating that further fine-tuning of term rendering is required (e.g. `vlaknast*ega* beton visoke trdnosti`). The observed problems with unithood are predominantly truncated candidates (e.g. `/?/ proces na` `mednarodne trge`), candidates spanning across the border of a term have not been observed in the analysed sample.

Finally, the best-performing category regarding termhood of the extracted candidates were those that contain conjunctions (75% wrt. 60% candidates with prepositions and 58% for adjective+noun combinations). False positives are either phrases common in general-language (e.g. `pogovor o likovni nalogi`) or unusual constructions, specific of a particular thesis in the corpus (e.g. `cestni otrok v makejevki`).

Results for 3-grams. 3-grams were evaluated by examining 100 random candidates from the list of 1,000 top-ranking extracted term candidates. Adjective+noun combinations and candidates containing prepositions were equally prolific (43% wrt. 42%, e.g. `gostota magnetnega pretoka, računalništvo v oblaku`). While 15% of the candidates were verb phrases, it turns out they were all noise as they all contained the verb to be, so they were excluded from further analysis (e.g. `biti v uporabi, v raziskavi smo`). The term grammar needs to be refined accordingly.

Unithood and structural accuracy is better preserved in candidates containing prepositions (83% wrt. 70% in adjective+noun combinations) where the biggest problem seems to be the preservation of the gender and number of the premodifiers (e.g. `magnetn*e* poljsk*e* jakost`). Manual analysis gives clear indications that term candidates already subsumed in longer phrases should receive special treatment (e.g. `sistem za podporo *odločanju*` wrt. `sistem za podporo`).

Termhood, the toughest test for the extracted candidates, shows that 63% of the candidats containing prepositions could be considered terms while the rest are flormulae typical of academic writing (e.g. `razlika med anketiranci`) or even general-language constructions (e.g. `spoznavanje prek spleta`). Adjective+noun combinations do better in this respect, achieving 73% accuracy. Again, scholar-specific phrasing is frequent (i.e. `teza doktorske disertacije`), slightly less so as far as general-language patterns are concerned (e.g. `nova finančna storitev`). It is interesting to note that term candidates extracted from technical and natural sciences theses typically suffer from unithood issues while lack of termhood is generally observed in the candidates extracted from social sciences and humanities documents.

4 Conclusions

We presented the construction of the first version of the Sketch Engine term grammar for Slovene and its application to term extraction from a corpus of PhD theses from different scientific domains against the slTenTen corpus. In manual evaluation we focused on 4- and 3-grams for which we analysed pattern productivity, unithood and structural accuracy of the extracted candidates as well as their termhood. While substantially fewer 4-grams were extracted, their pattern range was greater then in 3-grams. Eventhough unithood and structural accuracy varied more in 4-grams and was also lower in general than in 3-grams, termhood results were similar in both. This suggests that accuracy can be easily improved by further refining the term grammar.

The presented term grammar for Slovene is applicable to other corpora using compatible morpho-syntactic tagging and will be made freely available on the website of the KAS project: `http://nl.ijs.si/kas/english/`. Apart from term grammar refinement we plan to perform a set of comparative analyses on domain-specific subcorpora as well as extend the performance test to less scientific but more prolific MA and BA theses. For this, we will enhance the term extraction output which will enable the user to switch term ranking by termhood or by term patterns as needed in order to be able to focus on a particular term pattern or term pattern family. A systematic comparison of term extraction recall and precision with the CollTerm tool [16] will also be performed.

Acknowledgments. The work described in this paper was funded by the Slovenian Research Agency within the national basic research project "Slovene Scientific Texts: Resources and Description" (J6-7094, 2014-2017), the COST Action "European Network of e-Lexicography" (IS1305, 2013-2017) and the Ministry of Education of the Czech Republic within the Lindat Clarin Center.

References

1. Kalin Golob, M., Stabej, M., Stritar Kučuk, M., Červ, G., Koprivnik, S.: Genre analysis: Jezikovna politika in jeziki visokega šolstva v Sloveniji. Založba FDV (2014)
2. Swales, J.: Genre analysis: English in academic and research settings. Cambridge University Press (1990)
3. Genre, A.: Language use in professional settings. Applied Linguistics and Language Study.) London: Longman (1993)
4. Logar, N.: Aktualni terminološki opisi in njihova dostopnost. (2013) 58–64
5. Logar, N.: Korpusna terminologija: primer odnosov z javnostmi. Trojina: zavod za uporabno slovenistiko; Založba FDV (2013)
6. Daille, B., Gaussier, É., Langé, J.M.: Towards automatic extraction of monolingual and bilingual terminology. In: Proceedings of the 15th conference on Computational linguistics-Volume 1, Association for Computational Linguistics (1994) 515–521
7. Heylen, K., De Hertog, D.: Automatic Term Extraction. Handbook of Terminology, Volume 1. John Benjamins Publishing Company (2015)
8. Vintar, Š.: Bilingual term recognition revisited the bag-of-equivalents term alignment approach and its evaluation. Terminology **16**(2) (2010) 141–158
9. Erjavec, T., Fišer, D., Ljubešić, N., Logar, N., Ojsteršek, M.: Slovenska znanstvena besedila: prototipni korpus in načrt analiz. In Erjavec, T., Fišer, D., eds.: Proceedings of the Conference on Language Technologies and Digital Humanities, Ljubljana, Slovenia, Academic Publishing Division of the Faculty of Arts (2016) 58–64
10. 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)

11. Kilgarriff, A., Jakubíček, M., Kovář, V., Rychlý, P., Suchomel, V.: Finding terms in corpora for many languages with the Sketch Engine. EACL 2014 (2014) 53
12. Dagan, I., Church, K.: Termight: Identifying and translating technical terminology. In: Proceedings of the Fourth Conference on Applied Natural Language Processing. ANLC '94, Stroudsburg, PA, USA, Association for Computational Linguistics (1994) 34–40
13. Logar, N., Špela Vintar, Špela Arhar Holdt: Luščenje terminoloških kandidatov za slovar odnosov z javnostmi. In: Proceedings of the Eighth Language Technologies Conference, Jožef Stefan Institute (2012) 135–140
14. Kilgarriff, A.: Comparing corpora. International journal of corpus linguistics **6**(1) (2001) 97–133
15. Jakubíček, M., Kilgarriff, A., Kovář, V., Rychlỳ, P., Suchomel, V., et al.: The tenten corpus family. In: 7th International Corpus Linguistics Conference CL. (2013) 125–127
16. Pinnis, M., Ljubešic, N., Stefanescu, D., Skadina, I., Tadic, M., Gornostay, T.: Term extraction, tagging, and mapping tools for under-resourced languages. In: Proceedings of the 10th Conference on Terminology and Knowledge Engineering (TKE 2012), June. (2012) 20–21

Large Scale Keyword Extraction using a Finite State Backend

Miloš Jakubíček[1,2], Pavel Šmerk[1]

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

[2]Lexical Computing
Brighton, United Kingdom and Brno, Czech Republic
{milos.jakubicek}@sketchengine.co.uk

Abstract. We present a novel method for performing fast keyword extraction from large text corpora using a finite state backend. The FSA3 package has been adopted for this purposes. We outline the basic approach and present a comparison with previous hash-based method as used in Sketch Engine.

Key words: terminology extraction, keyword extraction, fsa, Sketch Engine

1 Introduction

In this paper we focus on the keyword (and terminology, as explained later) extraction task when solved using a system with a contrastive approach, such as the Sketch Engine corpus management system [1]. In this case, the input for this task consists of two arbitrary corpora: a focus corpus from which the keywords should be extracted, and a reference corpus that the term candidates from the focus corpus are contrasted with.

The keyword candidates come from different sources, but in the end the procedure always boils down to a very costly operation of matching all keyword candidates in the focus and reference corpus. While individual corpora are indexed in a database that assigns unique numeric identifiers to each string, hence intra-corpus processing operates on numbers and not strings, inter-corpus processing cannot take of this advantage and the any kind of pre-indexing (e.g. of particular corpus pairs) is not very flexible as systems like Sketch Engine deal with thousands of corpora, and the term extraction functionality is often used with user corpora built on-demand.

To speed up the process of string comparison, we present an approach that builds on intersecting two finite state automata. We show that this approach is more efficient both in space and time. We describe both the old method used in Sketch Engine and this new one and conclude by a comparison on a set of scenarios.

2 Keyword Extraction in Sketch Engine

Sketch Engine contains a keyword and terminology extraction module [2] using a contrastive approach to find term candidates. Two corpora are given as input to the term extraction: a focus corpus consisting from texts in the target domain, and a (ideally very big) reference corpus which the focus corpus is compared to. Sketch Engine currently contains reference corpora for over 80 languages.

The elementary units for the extraction can be one of the following three:

1. **positional attributes** in the corpus, such as word forms, lemmas or part-of-speech tags,
2. **terms** as identified by the language-specific term grammars, e.g. noun phrases,
3. **collocation lists** represented by triples of *(headword, relation, collocate)* as derived from the word sketches.

In each case the system first extract all candidates from the focus corpus so as to be able to compare their relative frequencies (or other statistic) with the reference corpus.

2.1 Previous approach

The previous approach as used in Sketch Engine was based on a string-to-string comparison in the case of positional attributes, and comparison of pre-indexed fixed-length string hashes in the case of term and collocation lists. Especially the latter case suffered from a number of deficiencies:

– pre-indexing of string hashes was costly both in space and time. E.g. for a English corpus enTenTen12 [3] which has almost 13 billion words, the collocation list hashes occupies 2.2 GB (each hash being a 64bit binary). This is a problem especially with a cold disk cache when the whole file needs to be read into the memory for any comparisons.
– even the comparison of hashes took a long time (e.g. the comparison of the collocation list between the British National Corpus [4] and the enTenTen12 still took about 2 minutes with a cold disk cache, and about 13 seconds with hot disk cache.)

2.2 Finite-state based approach

To overcome the disadvantages described above we have designed a new method based on finite state automata (FSA). Instead of pre-indexing any hashes, for all the three source types we pre-index a minimal FSA containing all the strings. We use the FSA3 package[1] which can efficiently build a minimal FSA and provides a string-to-number and number-to-string mapping of each

[1] See `http://corpus.tools`.

stored string (where the numbering corresponds to enumeration of all strings sorted lexicographically).

Initial experiments with FSA-based string-to-number and number-to-string mappings were described in [5]. The FSA3 package is inspired by the tools for automata generation and minimization developed by Daciuk [6]. Alongside of the development described below, we have significantly improved compile-time performance of the FSA3 package which is now about 10 times faster compared to what was provided in [5] and has linear complexity with regard to the input data size. While the compile-time performance is not crucial for the keyword extraction task (where the compilation is performed only once per corpus) it is an important aspect for other tasks where automata need to be often recreated. A detailed report on all findings relevant to the automata compilation and minimization will be provided in a separate paper.

We have extended the FSA3 package by the intersect operation on two automata (denoted as FSA_1 and FSA_2), which can:

1. output all strings present in both FSA_1 and FSA_2 together with their respective numeric IDs in both automata (we call this an **intersect operation**).
2. output all strings present in FSA_1 with matching numeric IDs or just the ID from FSA_1 where the string was not present FSA_2 (we call this a **left intersect operation)**.

The left intersect operation allows these automata to be directly exploited in the keyword extraction task so as to obtain a list of matching strings and IDs which can be used to retrieve pre-indexed frequencies from the individual corpora (where the string is present only in FSA_1, the frequency in FSA_2 is obviously zero).

3 Evaluation

We have conducted a number of comparisons of the hash-based and finite-state based approach using different usage scenarios and string sources. For the evaluation we used three corpora: the BNC (100 million words), the enTenTen12 (13 billion words) and enTenTen15 (30 billion words).

All results are summarized in Table 1.

The evaluation shows that the FSA-based approach is faster for all hot-cache scenarios. The slowdown for the cold-cache scenario was, after a detailed inspection, caused by the fact that the hash indices stored far less amount of data (ca 250 MB) because of filtering out items with frequency lower than 1 per billion words. Therefore, this comparison cannot be seen as representative.

4 Conclusions

In this paper we have presented a novel method for extracting keywords from very large (billion word) corpora that is based on finite-state machines. The

Table 1: Evaluation of hash-based and FSA-based keyword extraction

string source	**corpus**$_1$	**corpus**$_2$	**FSA**$_1$ **size**	**FSA**$_2$ **size**	**page cache**	**time**$_{prev}$	**time**$_{now}$	**speedup**
lemma	BNC	enTenTen15	556k items 4 MB	26,426k items 340 MB	cold	80.2s	51.4s	1.56x
					hot	6.3s	0.7s	9x
term list	Brown family	enTenTen12	320k items 4 MB	164,189k items 2 GB	cold	1m11s	2m10.2s	0.54x
					hot	4s	1.2s	3.3x

evaluation shows promising results so that the method is going to be adopted for the Sketch Engine corpus management system so as to be able to carry out practical results from a production environment.

Acknowledgments. The research leading to these results has received funding from the Norwegian Financial Mechanism 2009–2014 and the Ministry of Education, Youth and Sports under Project Contract no. MSMT-28477/2014 within the HaBiT Project 7F14047.

References

1. 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)
2. Kilgarriff, A., Jakubíček, M., Kovář, V., Rychlý, P., Suchomel, V.: Finding terms in corpora for many languages with the Sketch Engine. EACL 2014 (2014) 53
3. Jakubíček, M., Kilgarriff, A., Kovář, V., Rychlý, P., Suchomel, V.: The TenTen Corpus Family. International Conference on Corpus Linguistics, Lancaster (2013)
4. Leech, G.: 100 million words of English: the British National Corpus (BNC). Language Research **28**(1) (1992) 1–13
5. Jakubíček, M., Rychlý, P., Šmerk, P.: Fast construction of a word-number index for large data. RASLAN 2013 Recent Advances in Slavonic Natural Language Processing (2013) 63
6. Daciuk, J., Weiss, D.: Smaller representation of finite state automata. Theoretical Computer Science **450** (2012) 10–21

Evaluation of the Sketch Engine Thesaurus on Analogy Queries

Pavel Rychlý

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

Abstract. Recent research on vector representation of words in texts bring new methods of evaluating distributional thesauri. One of such methods is the task of analogy queries. We evaluated the Sketch Engine thesaurus on a subset of analogy queries using several similarity options. We show that Jaccard similarity is better than the cosine one for bigger corpora, it even substantially outperforms the word2vec system.

Key words: distributional thesaurus, analogy queries

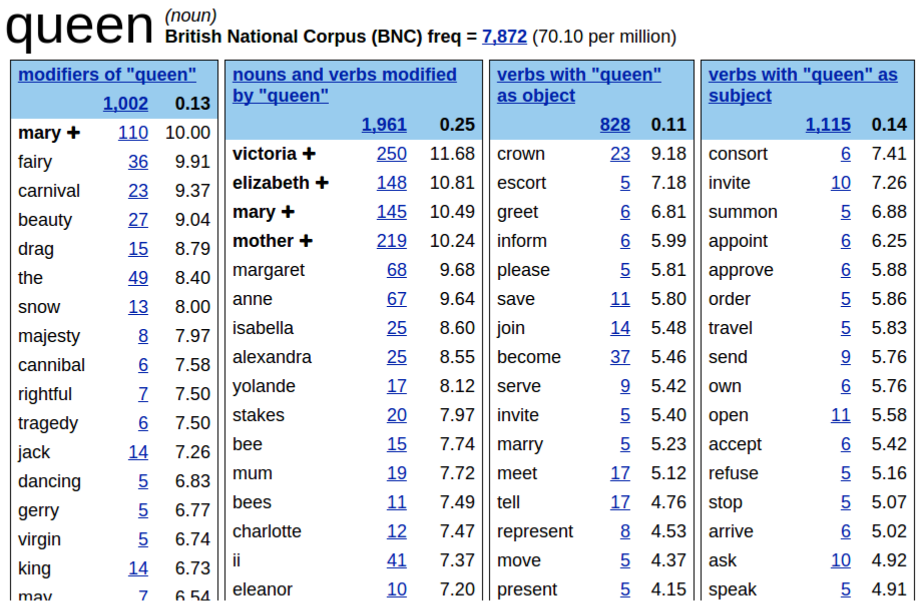
1 Introduction

A thesaurus contains words grouped together according to similarity of meaning. Like dictionaries, they are hard and expensive to produce. There is a long history of projects and tools trying to produce a thesaurus automatically from large text corpora. Such tools produce a list of similar words for each given word. Similar usually means words occurring in same or similar contexts. That could include not only synonyms and antonyms (as expected in human-created thesauri) but also words from the same class (like animals) or hypernyms and hyponyms. Such data sets (usually called distributional thesauri) are helpful for humans as another type of dictionary but they also useful in many natural language processing tasks.

There are many different approaches how to build a thesaurus from a text corpus with many parameters and options for each method. To compare which algorithm/settings is better there are methods for evaluating thesauri from the very beginning of building automatic thesauri in 1965 [1]. Thesaurus evaluation is discussed in more details in the next section.

The Sketch Engine (SkE) [2] is a corpus management system with several unique features. One of the most important feature (which also gave the name to the whole system) is a word sketch. It is a one page overview of grammatical and collocational behaviour of a given word. It is an extension of the general collocation concept used in corpus linguistics in that they group collocations according to particular grammatical relation (e.g. subject, object, modifier etc.). An example of word sketch for noun *queen* on British National

Corpus (BNC) [3] is in Figure 1. Another features of the Sketch Engine is a thesaurus. It is based on word sketches, similarity of two words is derived from the intersection of collocations in respective grammatical relations of both words. An example of the thesaurus result for noun *queen* on BNC is in Figure 2. More technical details of the Sketch Engine thesaurus computation are in Section 3.


queen (noun) British National Corpus (BNC) freq = 7,872 (70.10 per million)

modifiers of "queen"	1,002	0.13
mary +	110	10.00
fairy	36	9.91
carnival	23	9.37
beauty	27	9.04
drag	15	8.79
the	49	8.40
snow	13	8.00
majesty	8	7.97
cannibal	6	7.58
rightful	7	7.50
tragedy	6	7.50
jack	14	7.26
dancing	5	6.83
gerry	5	6.77
virgin	5	6.74
king	14	6.73
may	7	6.54

nouns and verbs modified by "queen"	1,961	0.25
victoria +	250	11.68
elizabeth +	148	10.81
mary +	145	10.49
mother +	219	10.24
margaret	68	9.68
anne	67	9.64
isabella	25	8.60
alexandra	25	8.55
yolande	17	8.12
stakes	20	7.97
bee	15	7.74
mum	19	7.72
bees	11	7.49
charlotte	12	7.47
ii	41	7.37
eleanor	10	7.20

verbs with "queen" as object	828	0.11
crown	23	9.18
escort	5	7.18
greet	6	6.81
inform	6	5.99
please	5	5.81
save	11	5.80
join	14	5.48
become	37	5.46
serve	9	5.42
invite	5	5.40
marry	5	5.23
meet	17	5.12
tell	17	4.76
represent	8	4.53
move	5	4.37
present	5	4.15

verbs with "queen" as subject	1,115	0.14
consort	6	7.41
invite	10	7.26
summon	5	6.88
appoint	6	6.25
approve	6	5.88
order	5	5.86
travel	5	5.83
send	9	5.76
own	6	5.76
open	11	5.58
accept	6	5.42
refuse	5	5.16
stop	5	5.07
arrive	6	5.02
ask	10	4.92
speak	5	4.91

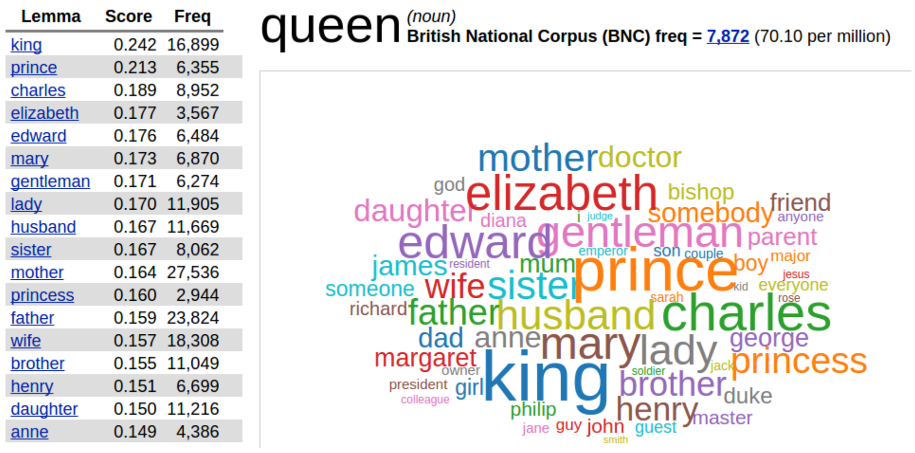
Fig. 1: Word Sketch of word *queen* on British National Corpus.

2 Thesaurus Evaluation

The first methods of evaluating thesaurus quality was based on gold standards – data prepared by several annotators. They contain a list of word pairs together with a numeric or quality assignment of their similarity. There are several problems with such data:

- some gold standards do not distinguish between similarity and relatedness (*money* – *bank*: score 8.5 out of 10 in WordSim353 data set [4])
- some gold standards do not provide any measure of similarity [5]

It is very hard for a human to decide which ordering of similar words is better. As an illustration, the Table 1 lists most similar words for noun *queen* from several sources.



Lemma	Score	Freq
king	0.242	16,899
prince	0.213	6,355
charles	0.189	8,952
elizabeth	0.177	3,567
edward	0.176	6,484
mary	0.173	6,870
gentleman	0.171	6,274
lady	0.170	11,905
husband	0.167	11,669
sister	0.167	8,062
mother	0.164	27,536
princess	0.160	2,944
father	0.159	23,824
wife	0.157	18,308
brother	0.155	11,049
henry	0.151	6,699
daughter	0.150	11,216
anne	0.149	4,386

queen *(noun)* **British National Corpus (BNC) freq = 7,872** (70.10 per million)

Fig. 2: Sketch Engine Thesaurus for word *queen* on British National Corpus.

Table 1: Most similar words for noun *queen* from different sources

Source	Most similar words to *queen*
serelex[5]	king, brooklyn, bowie, prime minister, mary, bronx, rolling stone, elton john, royal family, princess, monarch, manhattan, prince, harper, head of state, iron maiden, kiss, paul mccartney, abba, hendrix
Thesaurs.com	monarch, ruler, consort, empress, regent, female ruler, female sovereign, queen consort, queen dowager, queen mother, wife of a king
SkE on BNC	king, prince, charles, elizabeth, edward, mary, gentleman, lady, husband, sister, mother, princess, father, wife, brother, henry, daughter, anne, doctor, james
SkE on enTenTen08	princess, prince, king, emperor, monarch, lord, lady, sister, lover, ruler, goddess, hero, mistress, warrior, knight, priest, chief, god, maiden, brother
word2vec on BNC	princess, prince, Princess, king, Diana, Queen, duke, palace, Buckingham, duchess, lady-in-waiting, Prince, coronation, empress, Elizabeth, hrh, Alianor, Edward, King, bride
powerthesaurus.org	empress, sovereign, monarch, ruler, czarina, queen consort, king, queen regnant, princess, rani, queen regent, female ruler, grand duchess, infanta, kumari, maharani, crown princess, kunwari, shahzadi, malikzadi

With recent research on vector representation of words [6] they came new methods of evaluating such representations. One of such methods is the task of analogy queries. Each query is in form "a is to a^* as b is to b^*", where b^* is hidden and the system mush guess it. There are several types of analogy in the evaluation data set, we can divide them into two classes: morpho-syntactic ("*good* is to *best* as *smart* is to *smarter*") and semantic ("*Paris* is to *France* as *Tokyo* is to *Japan*"). Most of such queries are easy to answer by humans and there is almost 100 % agreement.

Using vector representation of words, a vector of real numbers is assigned to each word. The analogy query is answered by finding the closest word to the result of vector $a^* - a + b$. The distance of vectors is computed as the cosine similarity of two vectors:

$$cos(x,y) = \frac{v_x \cdot v_y}{\sqrt{v_x \cdot v_x}\sqrt{v_y \cdot v_y}}$$

where v_x and v_y are the respective vectors of words x and y. The analogy query is answered by:

$$\arg\max_{b^* \in V} cos(b^*, a^* - a + b)$$

3 Sketch Engine Thesaurus

As we mentioned in the first section the Sketch Engine thesaurus is based on word sketches. It is computed during corpus compilation. For each word, all words with similarity above a threshold are stored together with the similarity score. An efficient algorithm is used to compute the whole $N \times N$ matrix [7].

The question is whether we can use the vector arithmetics for analogy queries together with Sketch Engine thesaurus. At first sight we cannot use it because we have no vectors. But we can derive the vectors from word sketches. Each collocation in a grammatical relation could be one dimension of the vector. The association score of the collocation is the value of that dimension in the vector. But there are two main differences:

– The dimension of word vectors in word2vec system (and also in all others) is only 100 – 1000, the dimension of vectors from word sketches goes up to millions.
– Similarity in SkE thesaurus is computed using Jaccard similarity instead of the cosine one.

Fortunately, vector arithmetic could be interpreted in different way, using CosAdd [8]:

$$\arg\max_{b^* \in V} cos(b^*, a^* - a + b) =$$
$$\arg\max_{b^* \in V}(cos(b^*, a^*) - cos(b^*, a) + cos(b^*, b))$$

It means that we are finding word b^* that is close to a^* and b and far from a. We can also use multiplication instead of addition and use CosMul:

$$\arg\max_{b^* \in V} \frac{cos(b^*, a^*)cos(b^*, b)}{cos(b^*, a)}$$

In our experiments we have defined two other methods with the cosine similarity substituted to the Jaccard one: JacAdd, JacMul.

4 Evaluation

We made the evaluation on two English corpora: BNC and SkELL. BNC (around 100 million tokens) is rather small for semantic relations, SkELL [9] (around 1.5 billion tokens) is large enough. We have selected only one analogy relation (*capital-common-countries*) from the analogy data set because many of the words from other relations have small number of hits in our corpora.

The results are summarised in Table 2. All experiments were evaluated on 462 analogy queries. The table lists number of successful answers and the respective percentage (accuracy) for each configuration.

Table 2: Results on capital-common-countries question set

	BNC		SkELL	
	count	percent	count	percent
CosAdd	58	12.6	183	39.6
CosMul	99	21.4	203	43.9
JacAdd	32	6.9	319	69.0
JacMul	57	12.3	443	95.9
word2vec	159	34.4	366	79.2

5 Conclusions

The analogy queries is a very good task for evaluating distributional thesauri. Our results confirm the well-known fact that more texts provides better results. It seems that the cosine similarity gives better results than the Jaccard similarity (SkE default) for smaller corpora. For bigger corpora, the Jaccard similarity is better than the cosine one.

For smaller corpora, word2vec is clearly better option but Sketch Engine thesaurus substantially outperforms word2vec for bigger corpora.

Acknowledgments. This work has been partly supported by the Grant Agency of CR within the project 15-13277S.

References

1. Rubenstein, H., Goodenough, J.B.: Contextual correlates of synonymy. Communications of the ACM **8**(10) (1965) 627–633
2. Kilgarriff, A., Rychlý, P., Smrž, P., Tugwell, D.: The Sketch Engine. Proceedings of Euralex (2004) 105–116 `http://www.sketchengine.co.uk`.
3. Aston, G., Burnard, L.: The BNC handbook: Exploring the British National Corpus with SARA. Edinburgh University Press (1998)

4. Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., Ruppin, E.: Placing search in context: The concept revisited. In: Proceedings of the 10th international conference on World Wide Web, ACM (2001) 406–414
5. Panchenko, A., Morozova, O., Fairon, C., et al.: A semantic similarity measure based on lexico-syntactic patterns. In: Proceedings of KONVENS 2012. (2012)
6. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
7. 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
8. Levy, O., Goldberg, Y., Ramat-Gan, I.: Linguistic regularities in sparse and explicit word representations. In: CoNLL. (2014) 171–180
9. Baisa, V., Suchomel, V.: Skell: Web interface for english language learning. In: Eighth Workshop on Recent Advances in Slavonic Natural Language Processing. (2014) 63–70

How to Present NLP Topics to Children?

Zuzana Nevěřilová, Adam Rambousek

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

Abstract. This paper deals with important, but underestimated aspect of research – presenting to the general public. Amongst many outreach activities, Natural Language Processing Centre takes part in Masaryk University project of courses for children 9 to 14 years old. We describe specifics to think of when presenting NLP topics to children, and use cases of previous and planned courses.

Key words: research publicity, NLP research presentation

1 Introduction

Presenting science to the public is sometimes underestimated since it is not considered to be scientific. Nevertheless, good presentation to the public can attract new students, industry partners, or influence fundings.

In this paper, we present one particular case – presentation for children attending the Masaryk JUniversity[1]. Masaryk JUniversity offers courses for children between 9 and 14, its main aim is to attract potential students to the university. Natural Language Processing Centre (NLPC) cooperates in the project from its beginning. This paper presents the topic we presented during two seasons and a new topic we plan to present in the forthcoming season.

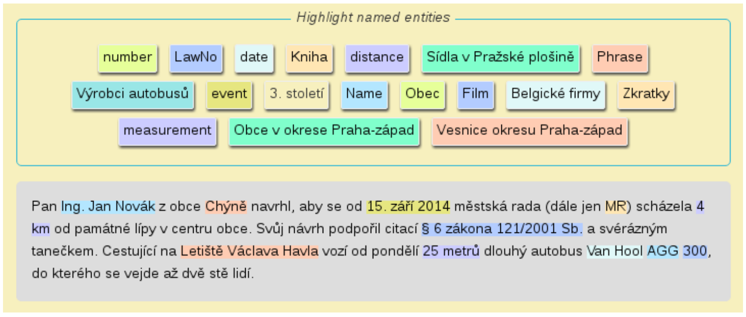
The result of the work is a short set of recommendations for future presentations of NLP to the public.

2 Presenting NLP research to general public

NLPC participates in various outreach activities to present results of NLP research and publicly available tools. These events may have different audience groups, e.g. high school students interested in computer science studies, or general public during European Researchers' Night. However, they usually do not have deep linguistic knowledge and do not realize the scope and meaning of NLP research.

Difficulty with linguistic and text processing research presentation is often the lack of physical object to demonstrate. The first important issue is to

[1] `http://www.mjuni.cz`



Pan Ing. Jan Novák z obce Chýně navrhl, aby se od 15. září 2014 městská rada (dále jen MR) scházela 4 km od památné lípy v centru obce. Svůj návrh podpořil citací § 6 zákona 121/2001 Sb. a svérázným tanečkem. Cestující na Letiště Václava Havla vozí od pondělí 25 metrů dlouhý autobus Van Hool AGG 300, do kterého se vejde až dvě stě lidí.

Fig. 1: Czech Named Entity Recognition with visually highlighted named entities.

Fig. 2: Word games during European Researchers' Night.

describe the tasks of NLP research and explain them using tools that people use in everyday life [1,2], e.g. spelling checkers, predictive typing, voice recognition, or machine translation. When presenting research results or NLP tools to public, try to use visually appealing graphical representation as much as possible. Not just to attract attention of the audience, but mainly because the information presented with the help of pictures enhance the chance of remembering [3,4]. For example, see visualization of named entity recognition in Figure 1.

During some events, visitors have opportunity to try our tools hands-on. For such occasion, we have created several computer games with the linguistic themes, e.g. finding the word by its related words, or computer asking questions to guess the player's word. Although not directly related to research in NLPC, we have created two language-related board games for visitors who like to play with words – semantic network with various word relations (see Figure 2) and modified crossword [2].

3 Presenting science to primary school children

Masaryk JUniversity offers courses for children aged between 9 and 14 years old and the capacity is usually limited to 170 participants. Thanks to the requirements, children who are interested in science and more keen to study in general are enrolled in the courses. On the other hand, age of 9 to 14 is still quite wide range, and skills and development of children may vary a lot [5,6].

Generally, there are several guidelines to use when preparing courses for children in preadolescent or early adolescent age [7,8]:

- Children are curious and not shy to ask, be prepared for a lot of questions. Sometimes even off-topic.
- Children are competitive and most of them like to move. Break the lecture after approximately 30 or 40 minutes and plan some team game when the children can get up.
- It is the age of changing emotionality, so try to avoid sensitive subjects.
- Do not forget that children do not share the same long-term experience. E.g. when we were presenting predictive writing tools, some children did not know keypad mobile phones.

When presenting NLP topics and tools to children, we have discovered several specific issues:

- Children happily destroy your software tools by simply trying what is possible, e.g. really large input values or unexpected options. As a general rule in software development, never trust user input and expect the unexpected.
- Some children are bilingual and do not realize it. For example, when asked for sample words, some children automatically responded in other languages than Czech.
- Speech recognition software have issues recognizing children's voices, because the recognition model is trained on adults.

4 Predictive writing presentation

In 2015 and 2016, we presented a well defined topic of NLP: predictive writing. The topic was selected since most users have personal experience with predictive writing, however, we assumed not many users thinking about the language technology behind the application.

The presentation outline was as follows:

[2] Original word ciphers from puzzle hunts Dnem and Krtčí norou, adapted to board games by Vít Suchomel.

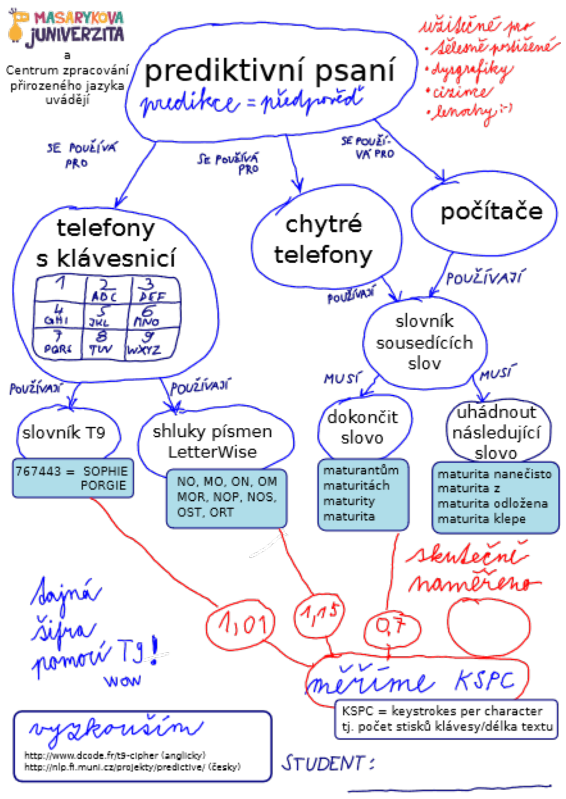


Fig. 3: Handout for predictive writing presentation.

- Predictive writing is something the audience is familiar with. We taught the term, we explained what can be predicted (word ending, next word, typo).
- Several systems exist depending on the hardware (restricted keyboards, touchscreens).
- The key software feature is a *dictionary* of words and/or n-grams and their *frequencies*.
- Physical activity: try to guess a word from a T9 typing.
- How to evaluate the quality of predictive writing software: keystrokes per character (KSPC, [9]), keystrokes on correction etc.
- Physical activity: In teams, try to compose as many words as possible from magnetic letters.
- Assistive technology: how can predictive writing help?
- Bonus material: switching among different user models in order to reduce KSPC.

We provide the audience a printed material (a single A4 sheet with a simple mindmap and links to applications, see Figure 3). We also let the children work with the Czech predictive writing demo[3].

4.1 Risks, pitfalls, and evaluation

We were unsure about the presentation intelligibility. The presentation was tested on one author's child, however, we would appreciate support from the university.

The web demo load was not tested before the presentation. The input length was not restricted. The web demo was running slowly because of the excessive web server load.

Some children were arguing that no predictive writing is needed if they used automatic speech recognition (ASR). However, no web demo contains children's voice models.

During presentation, we brought old mobile phones, since not all children have ever met mobile phones with keyboard.

The presentation was evaluated positively by the audience [10]. The topic was considered very easy or easy in two cases, just right in two cases. The presentation was considered intelligible, especially, the only comment to the presentation was "Surprisingly, I understood everything".

5 Semantic Network Presentation

For the year 2017, we have chosen a new topic. The main reason is that the number of children familiar with T9 prediction drops rapidly. The main risk we see is that unlike predictive writing, users do not have everyday experience with semantic networks.

The presentation outline is planned as follows:

– How to explain meanings of words to humans and to computers? *Connection* with other meanings is a key feature of all explanations (except of deixis).
– We need *memory* to store information about meanings, so do computers (analogy). In order to understand, we also need to know how to use information stored in memory: we present simple *entailments*.
– Semantic networks connect words that have something common in their meanings: same meaning, subclass, opposite meaning, has part etc. The connection itself explains the word meaning. Semantic network is both a way of storing information in memory, and a recipe to entail new knowledge.
– Physical activity: try to guess a word in our semantic network.
– Practical use of semantic networks: e.g. search engines use synonyms and hypernyms for query expansion.

[3] `https://nlp.fi.muni.cz/projekty/predictive`

– Bonus material: when entailments do not work. Explain monotonicity and the default rule [11].

We provide a web application demo that uses Czech WordNet [12] as a semantic network. We can show what can be entailed, and check whether it is true (manual annotation). We also present how an expanded query may improve the search results (e.g. on Google Images).

6 Conclusions

In previous sections, we presented shortly two presentations, the former already presented to children aged between 9 and 14, the latter planned for presentation in January 2017. Our experience lead to a short list of recommendations that is intended to help other scholars to present NLP research and topics to the general public. Presentation to children have few specifics compared to presentation to adults: children have less (long term) experience, they appreciate physical activity, and they are often like to compete.

7 Recommendations on presenting NLP to the public

1. Choose an appropriate subtopic of NLP, do not assume that the audience knows the field. You can present arbitrary topics, even a "difficult" ones if you are able to explain the essence of it in one simple sentence. For example: Predictive writing in cell phones is based on frequencies of words and word sequences.
2. Simplify the terminology and stay consistent in it. Do not use linguistic terminology (use sentence composition instead of syntax, word sequence instead of n-grams etc.). Use metaphors and analogies.
3. Explain words that mean something else in your field than in the general domain (e.g. *dictionary* in the general domain contains lemmata, dictionary in NLP means list of word forms)
4. Test the presentation before you give it. Let the test person(s) interrupt your test presentation with questions, comments, and associations they make.
5. Design an attractive an easy-to-use UI to your tools:
   – Language of the UI consistent to the language you used
   – Provide the UI for later use if possible
   – Log all activity, let the audience give a feedback if possible
6. Offer physical activity, specifically when presenting to children. Use physical objects, not only software.
7. Provide printed material.
8. Offer bonus material.
9. Let one main message (explanation) to emerge from your presentation
10. Expect off-topic questions and prepare answers to them. In language technology presentations, people often ask about orthography, language learning. In predictive writing presentations, people ask about speech technologies. In semantic networks presentations, people ask about artificial intelligence, machine translation, philosophy.

Acknowledgments. This work has been partly supported by the Ministry of Education of CR within the LINDAT-Clarin project LM2015071.

References

1. Purnell, T., Raimy, E., Salmons, J.: Making linguistics matter: building on the public's interest in language. Language and Linguistics Compass **7**(7) (2013) 398–407
2. McKee, C., Zimmer, E., Fountain, A., Huang, H.Y., Vento, M.: Public outreach in linguistics: Engaging broader audiences. Language and Linguistics Compass **9**(9) (2015) 349–357
3. Kosara, R., Mackinlay, J.: Storytelling: The next step for visualization. Computer (5) (2013) 44–50
4. Carney, R.N., Levin, J.R.: Pictorial illustrations still improve students' learning from text. Educational psychology review **14**(1) (2002) 5–26
5. Zelazo, P.D.: The Oxford Handbook of Developmental Psychology, Vol. 1: Body and Mind. Volume 1. Oxford University Press (2013)
6. Macek, P., et al.: Adolescence. Portál (2003)
7. Charlesworth, R.: Math and science for young children. Cengage Learning (2015)
8. Hassard, J., Dias, M.: The art of teaching science: Inquiry and innovation in middle school and high school. Routledge (2013)
9. 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
10. Masarykova univerzita, Rektorát: Dětská MASARYKOVA jUNIVERZITA MjUNI: Zhodnocení 2. ročníku 2015/2016 (2016) Internal material.
11. Allen, J.: Natural Language Understanding (2nd ed.). Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA (1995)
12. Pala, K., Smrž, P.: Building Czech Wordnet. Romanian Journal of Information Science and Technology **2004**(7) (2004) 79–88

Subject Index

Author Index

RASLAN 2016

Tenth Workshop on Recent Advances in Slavonic Natural Language Processing

Editors: Aleš Horák, Pavel Rychlý, Adam Rambousek
Typesetting: Adam Rambousek
Cover design: Petr Sojka

Published by Tribun EU
Cejl 32, 602 00 Brno, Czech Republic

First edition at Tribun EU
Brno 2016

ISBN 978-80-263-1095-2
ISSN 2336-4289