RASLAN 2015
Recent Advances in Slavonic Natural Language Processing

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

RASLAN 2015

Recent Advances in Slavonic Natural Language Processing

Ninth Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2015
Karlova Studánka, Czech Republic, December 4–6, 2015
Proceedings

Tribun EU
2015

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 978-80-263-0974-1
ISSN 2336-4289

Preface

This volume contains the Proceedings of the Ninth Workshop on Recent Advances in Slavonic Natural Language Processing (RASLAN 2015) held on December 4th–6th 2015 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 2015 offers a rich program of presentations, short talks, technical papers and mainly discussions. A total of 14 papers were accepted, contributed altogether by 21 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 Marek Medved'. 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 2015 Karel Pala

Table of Contents

Part I

Stochastic Language Analysis

Style & Identity Recognition

Jan Rygl

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

Abstract. Knowledge of the author's identity and style can by used in the fight against forged and and anonymous documents and illegal actions in the Internet. Nowadays, there are many systems dedicated to solving stylometric tasks, but they are predominantly designed only for a specific task; they are used exclusively by their owners; or they do not natively support any Slavic languages.
Therefore, we present new open-source modular system *Style & Identity Recognition* (*SIR*). The system is designed to support any stylometric tasks with minimal efforts (or event by default) by combining dynamic stylometry features selection and prediction driven by input data labels. The system is free for non-commercial applications and easy to use, therefore it can be helpful for people dealing with threatening e-mails or sms, children forum protection against pedophiles and other tasks. Being customizable and freely accessible, it can be also used as a baseline for other systems solving stylometry tasks.
System combines machine learning techniques and nature language processing tools. It is written in Python and it is dependent on other open-source Python libraries.

Keywords: stylometry, authorship recognition, machine learning, open-source

1 Introduction

Organizations such as forensic expert bureaus, national security agencies and big companies are using advanced language tools to predict style and identity of author of the text.

But the tools which these organizations are using are predominantly limited by two factors:

1. **Exclusive rights**
   e.g. *ART*[1] [1] is used by Ministry of the Interior of CR,
   or *FLAIR*[2] is used only by a forensic expert bureau.

[1] *ART*: Authorship Recognition Tool

[2] *FLAIR*: Forensic Linguistic Advice, Investigation and Research, see `http://www.forensiclinguistics.net/`

2. **No support for other languages such as Slavic**
   e.g. *JGAAP*[3][2] lastly updated in 2013.

The main usage of the stylometry tools includes the following tasks:

1. authorship verification (court evidence; Internet authentication)
2. authorship attribution (criminal investigation of anonymous illegal documents)
3. authorship clustering (multiple authorship detection in essays, multiple accounts illegally used by one user [3])
4. age prediction (pedophile detection in children web communication [4])
5. translation detection (was the text translated by a person or by an automatic method?)
6. mental illness recognizer (detect symptoms and warn people)
7. personality analyzer (predict personality traits for human resources)
8. ...

Most of existing tools are specialized and not publicly available. Therefore, there is a place for an accessible free tool which can handle any stylometry task and can be used by non-expert users with almost optimal performance. Our goal is to fill this gap, therefore we present *Style & Identity Recognition* (*SIR*) tool.

2 Stylometry analysis

Author's style can be defined as a set of measurable text features (style markers) according to stylostatisticians [5]. Definition can be extended by adding non-text features such as colors, link domains and publication times.

The good example of style markers are frequencies of word-lengths. They were used as the first deterministic stylometry technique to detect an authorship of documents. T. C. Mendenhall discovered that word-length frequency distribution tends to be consistent for one author and differs for different authors (1887, [6]).

Style markers can depend on the properties of texts (formatting richness) and by tools which were used to extract them.

Modern methods use machine learning to process style markers extracted from documents. Machine learning techniques such as Support Vector Machines [7] and Random Forests always outperform pure distance metrics such as cosinus similarity (used for example in authorship verification).

3 Components of style & identity recognition

1. Stylometry corpus builder
2. Text cleaning (boiler-plate removal, HTML removal, etc.)

[3] *JGAAP*: Java Graphical Authorship Attribution Program, see `www.jgaap.com`

3. Language detection
4. Encoding detection
5. Text tokenization and further analysis
6. Semantic analysis (entity detection, abbrevation expansion, etc.)
7. Style markers selection
8. Style markers extraction
9. Machine learning processing

3.1 Stylometry corpus builder

Since we are using machine learning techniques, we need documents to tune features (style markers extractors), to train classifiers and to evaluate them. For English and other majority languages, there are many available language sources (e.g. e-mail corpus Enron [8]).

But for Slavic languages such as Czech and Slovak, there are several very small manually collected collections (e.g. Czech essays of pupils [9]). But there is also a current project *Authorship corpora builder* (*ACB*)[10] focused on small European languages. *ACB* contains free pre-built corpora for Czech and Slovak languages and tools for building new corpora. The tool and built corpora are freely accessible at `https://nlp.fi.muni.cz/projekty/acb/`.

Since there are existing tools and data sources, data collection is not planned to be part of *SIR* tool.

3.2 Text cleaning

Boiler-plate and markup removal phase is the most effective if it is done during the process of data crawling (we know the structure of the data domain and can compare an analyzed document with a big set of documents – even with documents not meant to pass data selection process).

Therefore, we use text cleaning already present in data sources and do not perform further text cleaning in *SIR* tool.

3.3 Language detection

Language detection is very important because style markers (machine learning features) depend on the language of documents. Features based on morphology, syntactic analysis or entity detection require to be given the language of a document before a document procession step.

Our system uses *langid* [11] library (`https://github.com/saffsd/langid.py`).

3.4 Encoding detection

Despite the fact that more than 85% of web pages use `utf-8` encoding [12], the encoding detection process can be still useful. Middle-European languages such as Czech and Slovak use non-ascii characters and bad encoding detection can negatively influence text post-processing (e.g. morphology analysis). We recommend to use Chared[4] library, our *SIR* tool natively supports it.

3.5 Text tokenization and further analysis

There are many libraries supporting naive text tokenization (word separation based on white spaces and punctuation). In language independent application, we need one robust general tokenizer usable for all languages. If we have a specialized tokenizer for given languages, it can be used instead of a general one.

As a general tokenizer, `nltk.tokenize.WordPunctTokenizer` is used. The *SIR* tool also supports morphology analysis using RFTagger [13]. If RFTagger is used, not only morphology tagging is performed for supported languages (Czech, Slovak, Slovene, German, Hungarian, Russian), but also tokenization is done by RFTagger.

In following versions, support for other morphology taggers and syntactic analyzers will be added. The taggers are not part of the project and they are used as external libraries instead because of licensing restrictions.

3.6 Semantic analysis

There are two reasons not to implement general multilingual semantic analysis:

1. For each language, we need implementation of one semantic analyzer (e.g. named entity in one language is not a named entity in another language).
2. Style markers usually use only small part of semantic analysis output, therefore it is better to make specialized standalone analysis for each style-markers extraction (which can be faster and more accurate than complex analysis).

We decided that semantic analysis should be part of style-markers extraction phase.

3.7 Style markers selection

Style markers are divided into two categories:

1. Language independent style markers (e.g. word length, sentence length, capitalization)
2. Language dependent (e.g. syntactic analysis)

Special case is a morphology analysis. We are using RFTagger which uses similar tagset for all supported languages, therefore some style markers can be language dependent, but support wide range of languages.

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

The quality and the utility of style markers depend on the type of solved problem. Different document lengths and tasks require different style markers, therefore it is recommended to experimentally select a subset of style markers and not to use them all [14].

Style marker selection is a semi-automatic step, only style markers supporting language of a document are preselected, but user can narrow the selection by filtering out categories of style markers not suitable for current problem.

3.8 Style markers extraction

For each category of style markers (e.g. word length is a category, `word length 1`, `word length 2`, ... are style markers), there is one python class. *SIR* tool includes several implementations of established style-markers categories and others will be added in future. Users are allowed to add new categories depending on their demand, each category is defined by:

- feature list (e.g. stopwords `game`, `tv`, `chat`, `facebook`)
- for each feature, a function converting processed document (text, morphology analysis, title, publication time) to a float number (e.g. if game in text: features[0] = "game" in text).

3.9 Machine learning processing

We use *scikit-learn* [15] library. Default machine learning algorithm is Random Forest Classifier, but in future versions we will support automatic classifier selection.

Classifier parameters are found using cross-validation on train data and native *scikit-learn* grid search.

All features are scaled to range $\langle 0, 1 \rangle$.

4 SIR

The project is developed under *Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License*[5].

The system is implemented in Python and uses following third party libraries: *gensim*, *ipython*, *numpy*, *requests*, *scikit-learn*, *scipy*, *smart-open*, *sqlitedict*, *xmltodict*, *langid*, *Flask*, *Flask-Cors*, *Flask-Mako*, *gunicorn*, *argparse*, *nltk*, and *chared*. It also supports morphological analyzer RFTagger [13].

The project is located on *GitHub* at url `https://github.com/janrygl/sir`. Online demo is available at `nlp.fi.muni.cz/sir`.

[5] `https://creativecommons.org/licenses/by-nc-nd/4.0/`

5 Evaluation

For evaluation purposes, we used reference corpus 2.0 of *Authorship corpora builder*[6]. The **authorship attribution** problem was solved: *Given a particular sample of text known to be by one of a set of authors, determine which one* [2, p. 238].

We used Czech documents from the reference corpus and tested scenarios with 2, 5, 10, 15, 20 and 28 authors. To be objective, we ran tests for each candidate count 100times (except the highest 28 authors), each time randomly selecting different authors. Resulting accuracies and standard deviations are displayed in Table 1 and in Figure 1.

Table 1. Authorship attribution experiment.

Author count	Accuracy	Baseline	Iterations
2	84% ± 16%	50.00%	100 iterations
5	60% ± 15%	20.00%	100 iterations
10	49% ± 10%	10.00%	100 iterations
15	42% ± 8%	6.67%	100 iterations
20	39% ± 6%	5.00%	100 iterations
28	41% ± 0%	3.57%	1 iterations

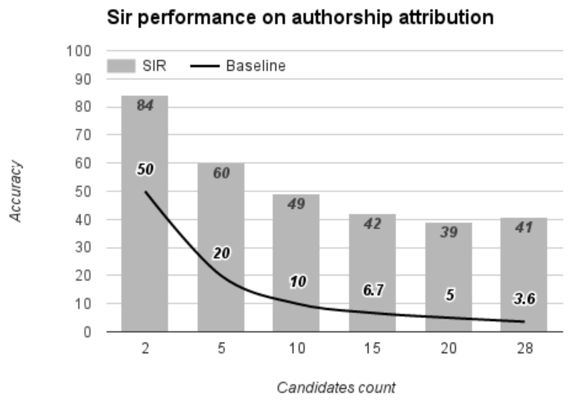


Fig. 1. Authorship attribution: results vs baseline.

[6] `https://nlp.fi.muni.cz/projekty/acb/getfile?name=download/author_corpus_v2.zip`

With growing number of candidates, accuracy is decreasing, but experimental results indicate that achieved accuracies are reasonably high and stable (reasonable standard deviation).

6 Conclusions and future work

We have introduced universal stylometric system ready to analyze documents. System can be downloaded from `https://github.com/janrygl/sir`.

We are going to actively develop system and add new features. Our plan is to provide accessible system that can by used by common Internet users to help them solve their stylometric tasks such as authorship attribution and gender recognition.

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

References

1. Rygl, J.: Art: Authorship recognition tool (2014)
2. Joula, P.: Authorship Attribution. Foundations and Trends in Information Retrieval. (2008)
3. Verhoeven, B., Daelemans, W.: Clips stylometry investigation (csi) corpus: A dutch corpus for the detection of age, gender, personality, sentiment and deception in text. 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) 3081–3085 ACL Anthology Identifier: L14-1001.
4. Peersman, C., Daelemans, W., Van Vaerenbergh, L.: Predicting age and gender in online social networks. In: Proceedings of the 3rd International Workshop on Search and Mining. SMUC '11, New York, NY, USA, ACM (2011) 37–44
5. Stamatatos, E., Fakotakis, N., Kokkinakis, G.: Automatic text categorization in terms of genre and author. Computational Linguistics **26**(4) (Dec 2000) 471–495
6. Mendenhall, T.C.: The characteristic curves of composition. The Popular Science **11** (1887) 237–246
7. Koppel, M., Schler, J., Argamon, S.: Computational methods in authorship attribution. J. Am. Soc. Inf. Sci. Technol. **60** (January 2009) 9–26
8. Klimt, B., Yang, Y.: Introducing the enron corpus. In: CEAS 2004 - First Conference on Email and Anti-Spam, July 30-31, 2004, Mountain View, California, USA. (2004)
9. Šebesta, K., collective of authors from ÚČNK FF UK: SKRIPT2012: akviziční korpus psané češtiny – přepisy písemných prací žáku základních a středních škol v ČR (in English: acquisition corpus of Czech written language – transcripts of the written work of pupils in primary and secondary schools in the Czech Republic) (2013)
10. Švec, J., Rygl, J.: Slavonic corpus for stylometry research. In: Proceedings of Ninth Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2015., Tribun EU, 1st ed. Brno (Czech Republic) (2015)

11. Lui, M., Baldwin, T.: Langid.py: An off-the-shelf language identification tool. In: Proceedings of the ACL 2012 System Demonstrations. ACL '12, Stroudsburg, PA, USA, Association for Computational Linguistics (2012) 25–30
12. W3Techs: http://w3techs.com/technologies/details/en-utf8/all/all
13. Schmid, H., Laws, F.: Estimation of conditional probabilities with decision trees and an application to fine-grained pos tagging. In: Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1. COLING '08, Stroudsburg, PA, USA, Association for Computational Linguistics (2008) 777–784
14. Rygl, J.: Automatic Adaptation of Author's Stylometric Features to Document Types. In: Text, Speech and Dialogue - 17th International Conference. 8655., Brno: Springer, 2014. (2014) 53–61
15. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12** (2011) 2825–2830

Slavonic Corpus for Stylometry Research

Ján Švec and Jan Rygl

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

Abstract. Stylometry techniques such as authorship recognition, machine translation detection and pedophile identification are daily used in applications for the most widely used languages. But under-represented languages lack data sources usable for stylometry research. In this paper, we propose an algorithm to build corpora containing meta-information required for stylometry experiments (author information, publication time, document heading, document borders) and introduce our tool *Authorship Corpora Builder* (ACB). We modify crawling and data-cleaning techniques for purposes of stylometry field and add heuristic layer to detect and extract meta-information.
The system was used on Czech and Slovak web domains to build a *Slavonic corpus* for stylometry research. Collected data have been published and we are planning to build collections for other languages and gradually extend existing ones.

Keywords: stylometry, slavonic corpus, web structure detection, corpora building

1 Introduction

Internet users are daily confronted with hundreds of situations in which they could use stylometry techniques. These situations include authorship detection (anonymous threats, false product reviews [1,2]); age and gender recognition (pedophile detection [3]); and spam and machine translation classification.

For dominant languages, there are many valuable data sources which can be used for a stylometry research (e-mail corpus Enron [4], age and gender corpus of Moshe Koppel [5]). Existence of these data sources enables fast implementation and comparison of the best techniques and facilitates further research.

But classic corpora (e.g. for Czech [6] and Slovak [7]) are unsuitable for several reasons:

– meta-information is missing (we do not know genres and publication times of texts; authors' identities, ages and genders);
– document borders are unclear;

– formatting is omitted (stylometry can use typography which cannot be witnessed in vertical corpus format)[1].

The lack of data for under-represented languages (such as languages of Visegrád Four) slows down development of stylometry tools for these languages. Therefore, we should contribute to building stylometry data sources before conducting further stylometry experiments in minor languages.

We propose a novel approach to building internet stylometry corpora and a modular system for collecting document with meta-information suitable for stylometry research is described. There are many systems for document crawling and text extraction but they are predominantly used for general corpora building (deduplication, boiler-plate removal, . . .). Selecting the most suitable algorithms and adding a layer of heuristic enable fully automated data acquisition. Our data are automatically annotated using information from web site. Documents without meta-information are omitted.

Having data with meta-information is the key to reliable results in computational stylometry. In this paper, we present a system, which was successfully used to build the Slavonic stylometry corpus[2], a freely available Czech and Slovak corpus that can be used for stylometry research and many other applications. We are also planning to build collections for other minor Slavonic languages and publish them.

2 Building a Slavonic corpus

Building a stylometry web corpora based on internet articles consists of downloading data from web (predominant crawlers can be used); detecting the structure of the web page (classic algorithms are optimized for boiler-plate removal; we need to modify them); text extraction (we modify text processing to keep valuable information for stylometry); and novel heuristic evaluation of extracted data.

Stylometry corpora differ from classic corpora by giving more emphasis on relationship between author and his documents. It is focused on documents, which are associated with certain author, so we can simply get various information from it. We can determine what vocabulary is the author using, how is he constructing sentences etc.[8]

2.1 Downloading data from web

A web crawler is used to gather all pages needed for further extraction process. We have implemented our crawler using slightly modified *Crawler4j*[3] java library. In our approach, we modify crawler to work with specific category on selected domain. Our work is focused on web domains built on template, which

[1] word-per-line (WPL) text, as defined at the University of Stuttgart in the 1990s

[2] Data are available at `http://nlp.fi.muni.cz/projekty/acb/preview`

[3] Crawler4j, `https://github.com/yasserg/crawler4j`

can be analyzed. We use a filter on URL for category, to ensure that the template on downloaded pages within one domain is the same.

2.2 Site Style Tree algorithm

Site style tree can be used for cleaning web pages. It is based on analysis of templates of HTML pages. It assumes that the important information on the page differs in content, size and shape opposite to non-important parts which have the same structure among many pages on the same domain.

It uses a structure called *Style Tree*. Pages are first parsed into DOM tree and then transferred to *Style Tree* that consists of two types of nodes, namely, *style nodes* and *element nodes*.

Style node represents a layout or presentation style and it consists of two components. First is a sequence of element nodes, second is number of pages that has this particular style. *Element node* is similar to node in DOM tree, but differs in his pointer to child nodes – which in element node is set on sequence of style nodes. Interconnection of *style node* and *element node* creates *Style tree* [9]. Example of Site Style Tree can be seen on fig. 1.

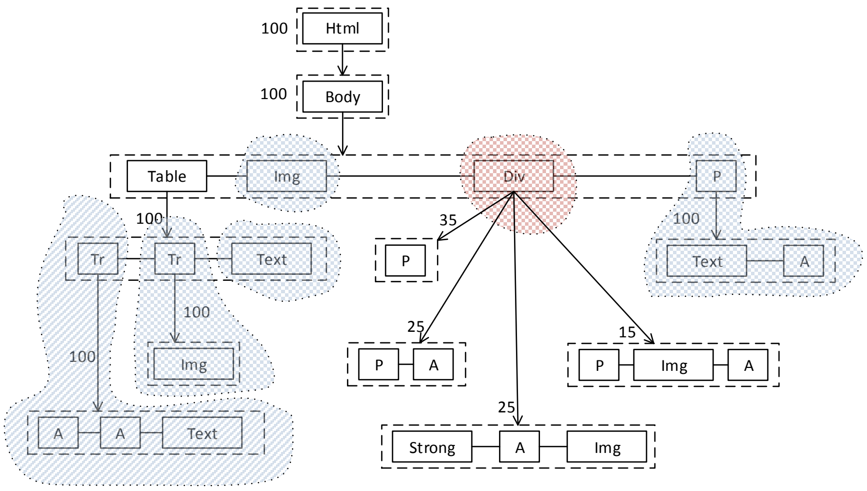


Fig. 1. Site Style Tree build from 100 pages.

In the SST we determine the important nodes, which have informational value like this: The more different child nodes *element node* has, the more important it is and, vice versa. We use weight for this attribute, which value can be between 0 and 1.

We used a metric, which measures the importance of element node [10]:

$$NodeImp(E) = \begin{cases} -\sum_{i=1}^{l} p_i \cdot log_m \cdot p_i & if\ m > 1 \\ 1 & if\ m = 1 \end{cases} \tag{1}$$

For particular element node E:

m is number of pages containing E,
l is number of child style nodes of E,
p is number of pages of particular child style node of E.

From the equation we can see that when E contains little child nodes, the value of NodeImp(E) will be small, and vice versa. For example – we can count NodeImp for Table (from fig. 1), like this:
$-0.35 \log_{100} 0.35 - 2 * (0.25 \log_{100} 0.25) - 0.15 \log_{100} 0.15 = 0.292 > 0$.

Computed value is afterwards compared with threshold. We can set the threshold in settings of our application. When the value is lower than threshold, element node is marked as non–informative, and vice versa.

2.3 Modifying SST for Stylometry Research

For our purpose we modified the SST algorithm in according way. If the input data contains a portal, which is unique between the pages, due to the *NodeImp* equation it receives value of one. Because we need to distinguish pages based on the same template and remove the boilerplate, portal pages and various pages which are very different are not the subject of our interest. We decided to remove unique pages from our analysis. For our needs we marked these pages as non–important. For increasing the efficiency of our algorithm, we explicitly increased the *NodeImp* value to nodes which may include the title, author or date, ensuring that these elements are kept. We use this approach to remove boilerplate from each web page. Afterwards it contains main text with meta data of the article - author, date and title.

2.4 Finding the article text

From modified SST we obtain only main text with meta-data. Text can contain boilerplate text – parts such as *a share button*, *a rate the article button*, *a link to comments*, etc. We can safely remove common text parts which are shorter than 20 characters[4] and doing so, boilerplate is removed.

Last step is to remove meta-data within the main text – author, date and title. When the required tags are found, we delete these tags from the main content of the page[5]. After the deletion we have only clean text of the article, which is stored.

[4] We have experimentally set the size limit to 20 after analysis of 5 different data sources
[5] More detailed description is in next subsections

2.5 Finding the author

ACB tries to find an expression "author(s)" in attributes of every tag in Slovak, Czech and English language (also tag `<author>` is checked). The score of each candidate tag is counted to determine the author (content must in most of cases abide name rules – regular expression preferring two words starting with capital letter). Because our current work aimed on small Slavic languages, we can detect female surnames by searching suffixes "ová". If tag with attribute "author" is not found, the algorithm tries to find the author in whole page. For example, we can look for name context such as *written by, published by, author*, etc. It is also important that the author is found near article text node in the SST, so the closer it is the higher score it has.

2.6 Finding the title

Title in most cases can be found in one of three different places:

1. tag `<h1></h1>` (or less probably in `<h2>`, `<h3>`, ...;
2. tag `<title></title>`;
3. attribute "title" of tag `<meta/>`.

A variant with the most diverse content (most of documents should have different titles) and abiding size limits (experimentally set min and max text length limits) is selected. If a found title contains boilerplate, e.g. *Server name: title name*, common phrase is automatically removed from the beginning of text.

2.7 Finding the date

In all examined languages, the algorithm searches an expression "date" between attributes of HTML tag. If we find this expression between attributes of a HTML tag, we check the content of this tag by regular expression recognizing date format (we also check relative times such as *today*, *yesterday*, ...

If there are multiple dates in one article, algorithm prefers the tag, which contains the attribute "date" and its contents corresponds to regular expression. It also prefers when date is found near article text in SST. If algorithm cannot find the expressions such as "date", it looks for all texts corresponding to date regular expression. If the matching text is found, its value is saved.

3 Results

We built a *Slavonic corpus* consisting of 15 Czech and Slovak web sites, from each we extracted 2000 articles. Therefore created corpus contains 30 000 documents and we plan to increase that number gradually. Collected data is published on `https://nlp.fi.muni.cz/projekty/acb/`[6].

[6] Data can be found in results section as version 2.

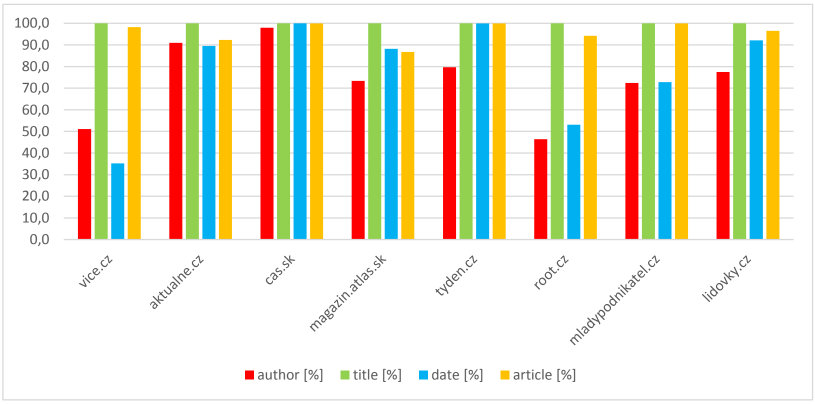


Fig. 2. Success rate of the first 8 sites

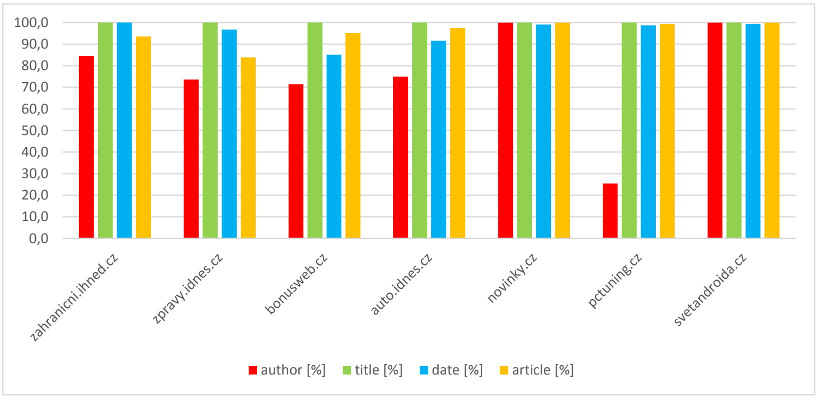


Fig. 3. Success rate of next 7 sites

We decided to use accuracy metric to determine the quality of acquired data (correctly found value is success, other cases are considered to be failure). Outside of scope of evaluation, we want to omit pages without found meta-information – portals, help pages, ...

We randomly selected 2000 documents and manually annotated them. Four categories were observed: author of the article, title, date and main text. If a correct author was found on 1800 pages from 2000, authors' accuracy was 90%; date was found on 1900 pages from 2000, accuracy is 95%, etc. We count accuracy for each category and we also compute average for each category and for whole algorithm. Success rate of the algorithm can be seen on fig. 2 and fig. 3.

Fig. 2 and 3 imply that the algorithm is most accurate in finding document title. For other categories the results are varying. It means that in some cases

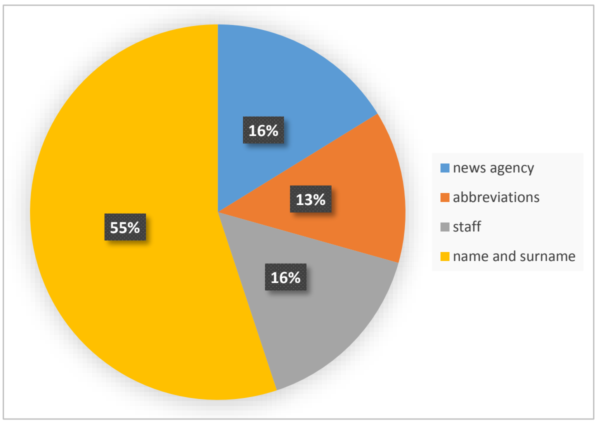


Fig. 4. Author data distribution

the important data was found only partly due to irregular structure of web pages. For example when it evaluated the page *vice.cz*, success rate of finding the date was 35.2%, because on most pages it uses different data format than we defined. It could be improved by extending the heuristic for finding date by adding more date formats.

Success rate for finding author is 74.6% , title is found in every case - 100%, date in 86,8% and article text in 95,8%. Average success rate of whole algorithm is 89.3%. Results indicate that algorithm described in this paper is suitable for building a *Slavonic corpus* for stylometry.

3.1 Authorship attribution

The **authorship attribution** problem can defined as follows: *Given a particular sample of text known to be by one of a set of authors, determine which one* [2, p. 238]. We selected authorship attribution for evaluation of our reference corpus because it is one of the most researched problems using a stylometry. The first deterministic stylometry technique was used to detect an authorship of documentsby T. C. Mendenhall in 1887 ([11]) and since then, authorship attribution is examined by security agencies, forensic experts, web authentication services and in many other tasks.

We used our *Slavonic corpus* as a training data source for authorship attribution. For our research the most important meta-information is author of the article, therefore if the algorithm don't find the author, the article is discarded – the corpora will always contain relevant data.

Correctly found authors (with their articles) are divided into 4 categories:

1. *news agency* - in our data mostly: "ČTK", "TASR" and "SITA",
2. *abbreviations* - author name in short form (two or three letters),
3. *staff* - article signed by name of web site, "staff" or "archive",

4. *name and surname* - full name of author.

Percentage of each category of our data is shown on fig. 4. For the problem of authorship attribution we focus only on data with authors' *name and surname* and omit other three categories.

In stylometry analysis, we want meta-information representing authors to correspond to one real person. Therefore, the 4th author category (full name of author is present) was used. Also, to define problem as having n candidates and trying to assign an author to an anonymous document, we want to document of each author to be present in tested data set to evaluate the task fairly. Therefore, we selected only authors with at least two documents, one document was used in a candidate set, once document was used in a test set of documents with examined authorship.

For each downloaded data set, we conducted following experiments:

1. We selected 2, 4, 8, 16 or 32 authors with at least 2 documents. If given set there wasn't enough documents, we took the highest possible count (e.g. in set with 10 documents, 2, 4, 8 and 10 candidates would be tested).
2. We divided documents to train (candidate) set and test (evaluation) set.
3. Documents were processed and analyzed using following stylometry techniques: word-length frequencies, stop-words frequencies, punctuation n-grams frequencies.
4. For each data set and candidate count, accuracy was measured and baseline established (baseline is $\frac{1}{\text{number of candidate authors}}$). Results are shown in Tables 1–7.

Table 1. atlas.sk

Author count	True	False	Accuracy	Baseline
2	25	1	96.15%	50.00%
4	26	4	86.67%	25.00%
8	38	40	48.72%	12.50%
14	53	61	46.49%	7.14%

Table 2. cas.sk

Author count	True	False	Accuracy	Baseline
2	4	0	100.00%	50.00%
4	5	3	62.50%	25.00%
8	9	11	45.00%	12.50%
16	11	28	28.21%	6.25%
32	12	74	13.95%	3.12%

Table 3. root.cz

Author count	True	False	Accuracy	Baseline
2	2	1	66.67%	50.00%
4	3	3	50.00%	25.00%
8	3	9	25.00%	12.50%
13	12	11	52.17%	7.69%

Table 4. svetandroida.cz

Author count	True	False	Accuracy	Baseline
2	5	1	83.33%	50.00%
4	13	7	65.00%	25.00%
8	20	23	46.51%	12.50%
16	21	58	26.58%	6.25%
19	29	65	30.85%	5.26%

Table 5. tyden.cz

Author count	True	False	Accuracy	Baseline
2	8	0	100.00%	50.00%
4	7	5	58.33%	25.00%
8	14	11	56.00%	12.50%
16	13	26	33.33%	6.25%
20	15	33	31.25%	5.00%

Table 6. vice.cz

Author count	True	False	Accuracy	Baseline
2	3	3	50.00%	50.00%
4	7	5	58.33%	25.00%
7	10	11	47.62%	14.29%

Table 7. zpravy.idnes.cz

Author count	True	False	Accuracy	Baseline
2	3	1	75.00%	50.00%
4	6	1	85.71%	25.00%
8	7	11	38.89%	12.50%
16	19	15	55.88%	6.25%
32	22	53	29.33%	3.12%

4 Conclusions

In this paper we described an application *ACB* which was used to build *Slavonic corpus* for stylometry research. The corpus contains contains articles grouped by author for each web domain. Every article contains also reference to its source, title and creation date. Main strength of the application is that building a corpus is fully automatic process – user is only required to insert start URL, and optionally adjust the settings.

Conducted authorship attribution experiments show that for low number of candidates, authorship can be attributed very reliably using basic stylometric features. For these scenarios, experimental results can be used as a baseline for other authorship attribution algorithms. If other scientist compares their data on the same data set, the methods become comparable.

Simple stylometric analysis is not good enough for higher number of candidates, but authors of other methods are welcome to establish more accurate baseline for these data using their techniques. We also plan to prepare download and attribute archives in which documents will be already divided into candidates authors' examples and examined instances. Each set will be named and highest achieved accuracy for the set will be dynamically updated in archive meta data.

5 Future work

We are currently working on full support of collecting of web discussions and very short documents – sites with more than one article per HTML page. We also plan to increase the precision by adding new heuristics for searching meta-information. We could also easily add an module for searching new types of meta-information. We are now working on expanding our *Slavonic corpus* and extending heuristics for other languages. Our goal is to build the biggest collection of stylometry data for under-represented European languages.

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

References

1. Chaski, C.E.: Who wrote it? steps toward a science of authorship identification. National Institute of Justice Journal (1997) 15–22
2. Joula, Patrick: Authorship Attribution. Foundations and Trends in Information Retrieval. (2008)
3. Peersman, C., Vaassen, F., Asch, V.V., Daelemans, W.: Conversation level constraints on pedophile detection in chat rooms. In: CLEF 2012 Evaluation Labs and Workshop, Online Working Notes, Rome, Italy, September 17-20, 2012. (2012)
4. Klimt, B., Yang, Y.: Introducing the enron corpus. In: CEAS 2004 - First Conference on Email and Anti-Spam, July 30-31, 2004, Mountain View, California, USA. (2004)
5. Koppel, M., Schler, J., Argamon, S., Pennebaker, J.: Effects of age and gender on blogging. In: In AAAI 2006 Spring Symposium on Computational Approaches to Analysing Weblogs. (2006)
6. Suchomel, V.: Recent czech web corpora. In Aleš Horák, P.R., ed.: 6th Workshop on Recent Advances in Slavonic Natural Language Processing, Brno, Tribun EU (2012) 77–83
7. Medved', M., Jakubíček, M., Kovář, V., Němčík, V.: Adaptation of czech parsers for slovak. In Aleš Horák, P.R., ed.: RASLAN 2012 Recent Advances in Slavonic Natural Language Processing, Brno, Czech Republic, Tribun EU (2012) 23–30

8. Koppel, M., Argamon, S., Shimoni, A.: Automatically categorizing written texts by author gender (2003)
9. Deepa, R., Nirmala, D.R.: Noisy elimination for web mining based on style tree approach. International Journal of Engineering Technology and Computer Research (IJETCR) **3** (2013) 23–26
10. Yi, L., Liu, B., Li, X.: Eliminating noisy information in web pages for data mining. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '03, New York, NY, USA, ACM (2003) 296–305
11. Mendenhall, T.C.: The characteristic curves of composition. The Popular Science **11** (1887) 237–246

The Initial Study of Term Vector Generation Methods for News Summarization

Michal Rott

Institute of Information Technology and Electronics,
Studentská 1402/2, 461 17 Liberec, Czech Republic
michal.rott@tul.cz

Abstract. In this paper, I present initial study of new term vector generation methods. The Random Manhattan Indexing and the Skip-gram model were introduced as novel techniques of term vector generation with interesting features. The purpose of this study is to determine whether the methods are suitable for the Summec: A Summarization Engine for Czech. The Summec already use Heuristic, TF-IDF and Latent Semantic Analysis methods for news article summarization. I test quality of generated vectors on the Summec's evaluation set and compare them with existing summarization methods. The novel summarization methods perform by 2 % worse than the LSA method. The evaluation set contains 50 newspaper articles, each annotated by 15 persons. The ROUGE toolkit is used to compare generated summaries with the human references. The above-mentioned evaluation set and the Summec demo are available online at http://nlp.ite.tul.cz/sumarizace.

Keywords: Latent Semantic Analysis, Random Manhattan Indexing, Skip-gram Model, Vector Space Model, Automatic Summarization

1 Introduction

Two novel methods of term vector generation were introduced in 2014. The first one is the Skip-gram model (SGM). Tomas Mikolov introduced very interesting features of the SGM in his paper [1]. This method is able to model relations between words and use them for further analysis. This model gained a lot of attention and is frequently tested in many NLP tasks; such as word clustering [2]. The second method is the Random Manhattan Indexing (RMI). The RMI takes advantage of random vector generation and this leads to extremely quick vector generation. The authors presented its ability to detect similarity of wikipedia pages [3].

I have already tested these methods in task of newspaper articles clustering [4]. Both methods outperformed the Latent Semantic Analysis (LSA). The LSA is used as basis method of text vectorisation for different NLP tasks; e.g. text indexing [5] or summarization [6,7]. In this paper, I want to test performance of the novel methods in task of single-document summarization and compare them with already implemented methods.

Linguistic properties of Czech language complicate the use of aforementioned methods. Czech words have many forms and their analysis requires lemmatization. The free word order limits language modelling and evaluation of summarization methods. Use of bi-grams and larger n-grams is questionable for evaluational by n-gram co-occurrence. Therefore, I use uni-grams to evaluate implemented methods. The last issues of Czech is rich vocabulary and there are many words representing the same thing. The vector generation algorithm have to unnecessarily train vectors for semantically similar words. Czech thesaurus [8] can be used to minimize influence of synonyms. A preprocessing module of the Summec solves this issues for Czech language and provides limited support for other Slavonic languages.

2 VSM and Summarization

The Vector Space Model is a well mathematically formed construct. The computation of similarity of the documents is reduced to computation of distance between their vectors. The distance can be computed by several metrics. Manhattan metric and cosine similarity are commonly used to compare vectors in NLP.

The main issue of the use of VSM is generation of vectors. The methods were tested: Latent Semantic Analysis, Random Manhattan Indexing and Skip-gram model. The first method is described in paper [9]. The rest will be described further in this paper.

A news article is composed from sentences and they are describing topics of the article. There are two possible ideas how to perform summarization in VSM:

1. Extract sentences of the main topic.
2. Extract the most important sentences of all topics.

The first idea is useful for single-document summarization. A sentence with the longest vector contains probably the most important terms of the document. Lets call this sentence the main sentence of the document. Sentence vectors with similar directions as the main sentence describe the same topic.

For example, Fig. 1 represents a document with 5 sentences and the objective is to extract the two most important sentences. Sentence s1 is clearly the most important sentence (the one with the longest vector) and the most similar sentence to s1 is s2. Hence, this summarization method will extract sentences s1 and s2.

The second idea is more suitable for multi-document summarization or large documents with more topics. The method extracts sentences with the longest and the most distant vectors between each other. The method alternates sentence vectors after every sentence extraction. This can be done iteratively and is described as follows:

1. Compute vectors of sentences from their term frequencies

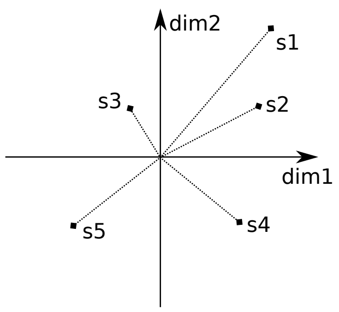


Fig. 1. Vector representation of document sentences in two-dimensional space.

2. Sentence with the longest vector is extracted.
3. The term frequency of words of the sentence are set to zero.
4. Repeat steps 1-3 and stop when extract is complete.

If we look at the Fig. 1, this method will extract sentence s1. After vectors alternation, s5 will be the longest vector. Sentences s1 and s5 will form the extract.

2.1 The Random Manhattan Indexing

The Random Manhattan Indexing method (RMI) was introduced in [3] and the main idea came from Random Projection. The LSA reduces dimensions using low rank approximation of space by Singular Value Decomposition. The main advantage of the RMI is the skip of SVD computation.

The RMI constructs L1 normed vector spaces with reduced dimensionality. It replaces the Euclidean metric with the Manhattan metric. The Manhattan Metric is not sensitive to non-Gaussian noise[1] [10]. Hence, the Manhattan metric yields good results in tasks of text similarity comparison.

The Manhattan metric is described as:

$$d(\boldsymbol{a},\boldsymbol{b}) = \sum_{i=0}^{N} |a_i - b_i| \tag{1}$$

where $\boldsymbol{a}$ and $\boldsymbol{b}$ are two vectors from a generated VSM and N is the number of dimensions.

The RMI is a two-step procedure. At first, index vectors are generated for terms of the text. Index vector $\boldsymbol{t}$ is randomly generated with the following probability distribution:

$$t_i = \begin{cases} \frac{-1}{U_1} & \text{with probability } \frac{s}{2} \\ 0 & \text{with probability } 1 - s \\ \frac{1}{U_2} & \text{with probability } \frac{s}{2} \end{cases} \tag{2}$$

[1] Non-Gaussian noise represents peaks in word frequencies. This phenomenon appears when a word is frequently repeated.

where U_1 and U_2 are two independent uniform random variables in (0,1) and s determines the sparseness of the index vectors. Value t_i represents i'th value of the index vector.

The sentence vectors are computed as the sum of index vectors. This sum is described as follows:

$$s_k = \sum_{t \in s} t_k \tag{3}$$

The k'th dimension of sentence vector s is computed as the sum of k'th dimension of every index vector of sentence s.

The problem with OOV words[2] is solved very simply; when a vector for a previously unobserved term is needed, a new vector is generated by probabilistic distribution (2).

2.2 The Skip-gram Model

The training of this model is very efficient and is based on log-linear neural network architecture. The training objective of the Skip-Gram Model is to find the best vector representation of terms in VSM that best predicts the surrounding words in the document.

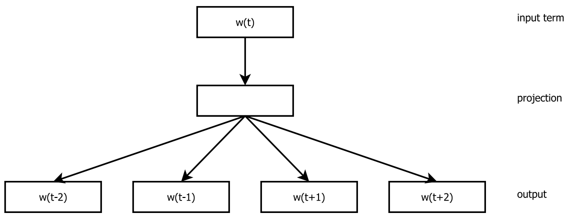


Fig. 2. The Skip-gram model architecture.

Formally, the objective is to maximize the average log probability for a given sequence of terms $t_1, t_2, ...t_N$

$$\frac{1}{N} \sum_{n \in N} \sum_{-c \leq j \leq c, j \neq 0} \log(p(t_{n+j}|t_n)) \tag{4}$$

where c is the size of the term context.

Computing log probability is not efficient because the cost of computing is dependent on the size of the training set. Therefore, the authors of the model define Negative sampling as objective which replaces the log probability computation with logistic regression [1].

[2] Out-of-Vocabulary words are words not observed in training data.

The Subsampling of Frequent Words is used to balance the occurrence of frequent terms (e.g., ″být", ″a", ″i" and ″v") with rarer terms that have more of an informational value. The terms′ probability is computed with the formula

$$P(t_i) = 1 - \sqrt{\frac{h}{f(t_i)}} \tag{5}$$

where $f(t_i)$ is the frequency of term t_i in the training data and h is the heuristically chosen threshold. The recommend value is around 10^{-5}.

The equation (3) is used to compute sentence vectors and cosine similarity (6) is used for vector comparison. The main disadvantage is handling of OOV words. The whole model supplemented by new data has to be recomputed to gain vectors for unobserved words. Nevertheless, the SGM produces very interesting spaces with semantically distributed word vectors as shown in [1].

$$d\left(\boldsymbol{a},\boldsymbol{b}\right) = \frac{\sum_{i=1}^{k} a_i \times b_i}{\|\boldsymbol{a}\| \times \|\boldsymbol{b}\|} \tag{6}$$

3 Experimental Evaluation

3.1 Data for Evaluation

There were no publicly available reference data for evaluation of Czech automatic summarization. Therefore, I created my own test set. This test set contains 50 newspaper articles gathered from Czech news servers. 15 people were asked to produce informative extracts of each article. The articles contained 92089 words in total and were selected from columns on local and international news, economics and culture. The reference extracts contain an average of six sentences. The evaluation set is available on the Summec web page.

3.2 Tools and Metrics Used

The ROUGE [11] was used for evaluation. This toolkit supports various metrics of summarization evaluation. I chose ROUGE-1 metric due to free word order of Czech language. The ROUGE-1 computes co-occurrence of unigrams in the reference and generated summaries. The results obtained using this metric are presented in terms of Recall, Precision and F-score:

$$Precision = \frac{TP}{TP + FP} \quad Recall = \frac{TP}{TP + FN} \quad F - score = \frac{2RP}{R + P} \tag{7}$$

where TP, FP and FN are explained in Table 1.

Table 1. The meaning of variables in equation (7) for ROUGE-1.

# unigrams	selected by anotators	not selected by annotators
selected by the system	TP	FN
not selected by the system	FP	TN

3.3 Experimental Setup

All summarization methods use preprocessing module of the Summec. This module offers sentence separation, words lemmatization, stop list, inverse document frequency dictionary and synonyms substitution. The separation of sentences is done using sequence of regular expressions that follows Czech language grammar. The Morphodita [12] is used to lemmatize the input texts. The stop list and IDF dictionary are created using 2.2M newspaper articles. The IDF dictionary contains 491k Czech lemmas. The resulting stop list contains over 200 Czech terms, including the most frequent Czech words and Czech prepositions, conjunctions and particles. Synonyms dictionary 7443 different groups of synonyms with a total of 22856 lemmas.

3.4 Comparison of Summarization Methods

The best performing method of the Summec (TFxIDF) and the LSA method are compared in 3.4 with novel methods. Both novel methods generated the exactly same extracts for our evaluation data. Hence, the result are identical.

Table 2. Comparison of ROUGE-1 score of summarization methods.

method	Recall [%]	Precision [%]	F-score [%]
LSA	55.4	55.1	55.2
RMI	50.7	56.7	53.3
SGM	50.7	56.7	53.3
TFxIDF	62.6	53.3	57.3

The novel methods scored by 1.9 % worse than the LSA and by 4 % worse than the TFxIDF. The result of the RMI methods was anticipated because this method uses random distribution to generate term vectors. By principle, these vectors can not semantically represent document sentences in VSM. In contrast, term vectors generated by the SGM method are represented semantically in space. Therefore, the same result as the RMI is surprising.

4 Conslusion

In this paper, I presented comparison of two news article summarization methods with methods implemented in the Summec. I evaluated their performance on an evaluation set containing 50 Czech news articles. The LSA-based method performs by 1.9 % better then the RMI and SGM methods. Nevertheless, the RMI offers very efficient way how to handle Out-of-Vocabulary words. The SGM has a feature of semantic representation of term vectors in VSM. I want to utilize this two methods in our next work. The demo of the Summec and the evaluation set is available on web page `http://nlp.ite.tul.cz/sumarizace` for public use.

Acknowledgments This paper was supported by the Technology Agency of the Czech Republic (project no. TA04010199) and by the Student Grant Scheme 2015 (SGS) at the Technical University of Liberec.

References

1. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations ofwords and phrases and their compositionality. In: Advances in Neural Information Processing Systems. (2013)
2. Lu, Y., Ji, D., Yao, X., Wei, X., Liang, X.: Chemdner system with mixed conditional random fields and multi-scale word clustering. Journal of Cheminformatics **7** (2015) cited By 3.
3. Zadeh, B.Q., Handschuh, S.: Random manhattan indexing. In: Proceedings - International Workshop on Database and Expert Systems Applications, DEXA. (2014) 203–208
4. M., R., Červa P.: Comparison of term vector generation methods for news clustering. In: 7th Language & Technology Conference, Poznan (Poland). (2015)
5. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE **41**(6) (1990) 391–407
6. Gong, Y., Liu, X.: Generic text summarization using relevance measure and latent semantic analysis. In: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. (2001)
7. Steinberger, J., Ježek, K.: Text summarization and singular value decomposition. In: Proceedings of the Third international conference on Advances in Information Systems. ADVIS'04, Berlin, Heidelberg, Springer-Verlag (2004) 245–254
8. Pala, K., Všianský, J.: Slovník českých synonym (Dictionary of Czech Synonyms, SČS). 3 edn. Lidove Noviny Publishers, Praha (2000)
9. Rott, M., Červa, P.: Summec: A summarization engine for czech. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **8082 LNAI** (2013) 527–535 cited By 0.
10. Weeds, J., Dowdall, J., Schneider, G., Keller, B., Weir, D.: Using distributional similarity to organise biomedical terminology. Terminology **11**(1) (2005) 107–141
11. Lin, C.Y.: Rouge: A package for automatic evaluation of summaries. In: Proceedings ACL workshop on Text Summarization Branches Out. (2004)

12. Straková, J., Straka, M., Hajič, J.: Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. In: Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, Baltimore, Maryland, Association for Computational Linguistics (June 2014) 13–18

Part II

Text Corpora

Towards Automatic Finding of Word Sense Changes in Time

Vít Baisa[1,2], Ondřej Herman[1], Miloš Jakubíček[1,2]

[1] Natural Language Processing Centre
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
{xherman1,xbaisa,jak}@fi.muni.cz

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

Abstract. We present a methodology proposal for finding changes in contextual behaviour of words. Assuming that a word sense is defined by actual usages of the word in a given context, this task corresponds to finding changes of word senses. We outline here main ideas of our distributional approach based on word sketches and discuss preliminary results.

Keywords: neologism, Sketch Engine, word sense, word sketch

1 Introduction

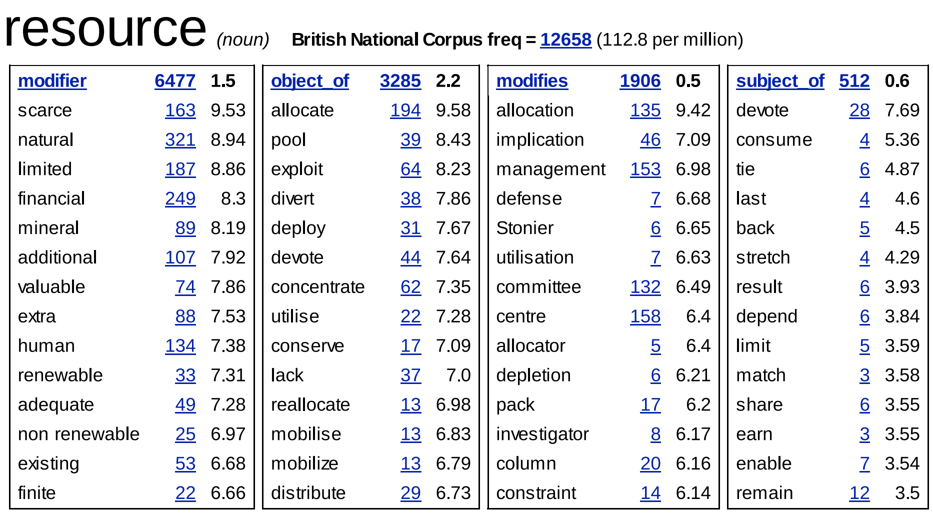
Thanks to technology advances, linguistics has undergone a major paradigm shift over the past 20 years—the field, previously largely dominated by an introspection, can now benefit from a vast amount of empirical evidence available in text corpora, incepting a new subfield—corpus linguistics. Most of the text corpora are synchronous—they represent a snapshot of some specific text type or general language at some time point. As such, they are suitable for many kinds of a synchronic analysis.

A diachronic analysis, i.e. tracking language changes over time, requires diachronic corpora carefully collected over large time spans, ideally with very balanced content (the same text types and text size from each time span). For English, some of the available corpora include the BNC [1] (which is however quite outdated and individual time spans are not very balanced), the COCA [2], the OEC [3] or the Feeds corpus [4]. A valuable resource, though not a text corpus, is also the Google Books n-grams database [5].

In this paper we discuss an extension to the existing trends finding functionality within Sketch Engine, a leading corpus management tool [6]. Sketch Engine has been recently [4] enhanced with a diachronic analysis function that tracks lexical changes in text corpora and estimates trends for individual words. This is useful also for neologism finding—however, many neologisms are not new word forms (lexemes) but new word senses. Building on the assumption that a word's sense is defined by the context in which the word is used, we propose a new approach (based on word sketches) for tracking changes of the context of a word which might often indicate a shift in its sense.

2 Sketch Engine

Sketch Engine is an online corpus management system providing access to hundreds of text corpora which can be searched and analyzed. It gained its name after one of its key features—word sketches, one page summaries of a word's collocational behaviour, in particular grammatical relations (see Figure 1). In technical terms, word sketches represent dependency syntax with each table item being a dependency triple consisting of a headword, a relation and a collocation, such as *resource–modifier–scarce*. These triples are scored using a lexicographic association score, in this case the logDice [7].

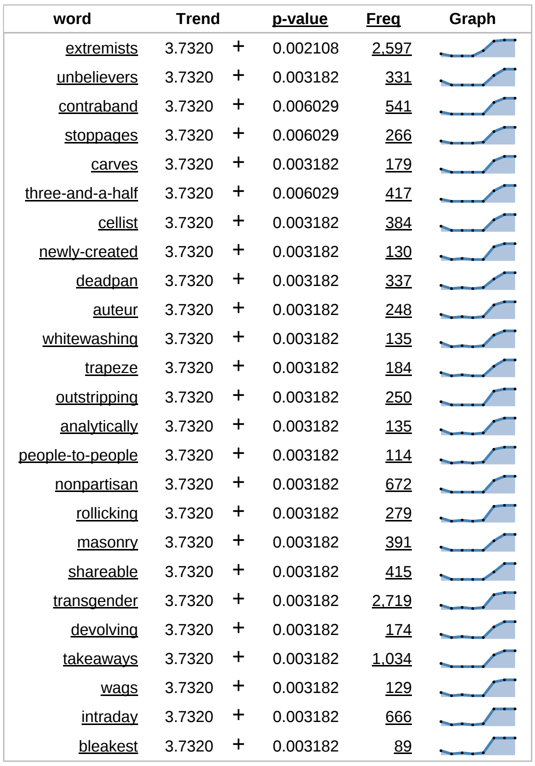


resource *(noun)* **British National Corpus freq = 12658** (112.8 per million)

modifier	**6477**	**1.5**	**object_of**	**3285**	**2.2**	**modifies**	**1906**	**0.5**	**subject_of**	**512**	**0.6**
scarce	163	9.53	allocate	194	9.58	allocation	135	9.42	devote	28	7.69
natural	321	8.94	pool	39	8.43	implication	46	7.09	consume	4	5.36
limited	187	8.86	exploit	64	8.23	management	153	6.98	tie	6	4.87
financial	249	8.3	divert	38	7.86	defense	7	6.68	last	4	4.6
mineral	89	8.19	deploy	31	7.67	Stonier	6	6.65	back	5	4.5
additional	107	7.92	devote	44	7.64	utilisation	7	6.63	stretch	4	4.29
valuable	74	7.86	concentrate	62	7.35	committee	132	6.49	result	6	3.93
extra	88	7.53	utilise	22	7.28	centre	158	6.4	depend	6	3.84
human	134	7.38	conserve	17	7.09	allocator	5	6.4	limit	5	3.59
renewable	33	7.31	lack	37	7.0	depletion	6	6.21	match	3	3.58
adequate	49	7.28	reallocate	13	6.98	pack	17	6.2	share	6	3.55
non renewable	25	6.97	mobilise	13	6.83	investigator	8	6.17	earn	3	3.55
existing	53	6.68	mobilize	13	6.79	column	20	6.16	enable	7	3.54
finite	22	6.66	distribute	29	6.73	constraint	14	6.14	remain	12	3.5

Fig. 1. An example of a word sketch for the noun *resource*.

Provided there is diachronic annotation available in a corpus, the system can be setup to offer a trends finding feature. This operates on individual corpus tokens whit associated positional attributes such as word form, lemma or part-of-speech tag. The frequency distribution of the input attribute value is subject to regression analysis (linear regression or Theil-Sen estimator) that estimates a trend defined by a curve slope. The user is then presented with an ordered list of most (positively or negatively) trending words as given in Figure 2.



word	Trend		p-value	Freq	Graph
extremists	3.7320	+	0.002108	2,597	
unbelievers	3.7320	+	0.003182	331	
contraband	3.7320	+	0.006029	541	
stoppages	3.7320	+	0.006029	266	
carves	3.7320	+	0.003182	179	
three-and-a-half	3.7320	+	0.006029	417	
cellist	3.7320	+	0.003182	384	
newly-created	3.7320	+	0.003182	130	
deadpan	3.7320	+	0.003182	337	
auteur	3.7320	+	0.003182	248	
whitewashing	3.7320	+	0.003182	135	
trapeze	3.7320	+	0.003182	184	
outstripping	3.7320	+	0.003182	250	
analytically	3.7320	+	0.003182	135	
people-to-people	3.7320	+	0.003182	114	
nonpartisan	3.7320	+	0.003182	672	
rollicking	3.7320	+	0.003182	279	
masonry	3.7320	+	0.003182	391	
shareable	3.7320	+	0.003182	415	
transgender	3.7320	+	0.003182	2,719	
devolving	3.7320	+	0.003182	174	
takeaways	3.7320	+	0.003182	1,034	
wags	3.7320	+	0.003182	129	
intraday	3.7320	+	0.003182	666	
bleakest	3.7320	+	0.003182	89	

Fig. 2. Most trending word forms in the Feeds corpus computed using the Theil-Sen estimation.

3 Finding Word Sense Changes

Sketch Engine identifies the most salient contexts in the word sketch feature. Therefore, we decided to experiment with using the existing trends computation on the top of word sketches. The input values for the regression analysis were frequency counts of the word sketch triples in the individual time spans. This way we obtained most trending triples which can be subject to further analysis (e.g. clustering of headwords). In Table 1 we list some linguistically interesting items found in the top 100 triples (both increasing and decreasing trends, sorted by the slope of the regression curve) identified in the COCA which covers a twenty-year period from 1990 until 2010, having 18 million tokens for each year (and additional 14 and 8 millions for 2011 and 2012).

Out of the top 100 increasing and top 100 decreasing triples only 19 are not noun-noun or adjective-noun modifiers. Clearly taking such triple list without further modification yields mainly trending multi-word noun phrases but not so many verb patterns. Therefore we also experimented with grouping the triples by headword. For each headword we calculated its score as the

Table 1. Sample trending triples (positive and negative) computed in the COCA.

+	−
peat-n modifies forest-n	percentage-n modifies plan-n
social-j modifies gradient-n	new-j modifies atheist-n
mastery-n modifies climate-n	diadromous-j modifies fish-n
joint-j modifies household-n	rate-n pp_of maltreatment-n
transaction-n modifies approach-n	medium-n modifier dual-j
trauma-n modifies theory-n	description-n pp_obj_for microfilm-n
learn-v object common-n	talk-v object pedometer-n
sexual-j and/or subjective-j	reverse-j modifies student-n
middling-j modifies sort-n	common-j modifies carriage-n
adult-j modifies day-n	new-j and/or packed-j
chick-n modifies lit-n	downwind-j modifies state-n
diminish-v subject regulation-n	cell-n object_of reprogram-v

multiplication of absolute values of all slopes in triples of this headword (which corresponds to composing the regression curves). Since the slope is often a small floating point number, we operate in an additive logarithmic calculus instead of directly multiplicative one.

The resulting scoring was clustered by part-of-speech of the headword into 4 groups: nouns, adjectives, verbs and others. In Tables 2–5 we provide the most changed headwords obtained together with the top 5 changed collocates—both positive ($\oplus$) and negative ($\ominus$):

4 Conclusions

We have presented an early stage research on novel methods for corpus-based finding of changes in word senses. The achieved results are promising but clearly indicate that more effort must be put into clustering of the word sketch triples and providing more insight into corpus data that would allow an easy interpretation of the results.

Acknowledgements This work has been partly supported by the Ministry of Education of CR within the Czech-Norwegian Research Pogramme in the HaBiT Project 7F14047.

References

1. Leech, G.: 100 million words of English: the British National Corpus (BNC). Language Research **28**(1) (1992) 1–13

2. Davies, M.: The Corpus of Contemporary American English as the first reliable monitor corpus of English. Literary and linguistic computing (2010)
3. Press, O.U.: Oxford english corpus (2015)
4. Kilgarriff, A., Herman, O., Bušta, J., Rychlý, P., Jakubíček, M.: DIACRAN: a framework for diachronic analysis. (2015)
5. Lin, Y., Michel, J.B., Aiden, E.L., Orwant, J., Brockman, W., Petrov, S.: Syntactic annotations for the google books ngram corpus. In: Proceedings of the ACL 2012 System Demonstrations. ACL '12, Stroudsburg, PA, USA, Association for Computational Linguistics (2012) 169–174
6. 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)
7. Rychlý, P.: A lexicographer-friendly association score. Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN (2008) 6–9

Table 2. Most trending nouns in COCA computed by grouping word sketch triples.

time-n	⊖constant-j, ⊖post-n, ⊖bat-n, ⊖unit-n, ⊕rating-n, ⊕hour-n, ⊕working-j, ⊕address-n
people-n	⊖distinguish-v, ⊖street-n, ⊖welfare-n, ⊖boat-n, ⊖wish-n, ⊕ensure-v, ⊕transgender-n, ⊕amazing-j, ⊕other-n, ⊕autism-n
year-n	⊖company-n, ⊖communism-n, ⊖recession-n, ⊖crucial-j, ⊕gender-n, ⊕collection-n, ⊕pound-n, ⊕multiple-j, ⊕old-j
man-n	⊖indigenous-j, ⊖advance-n, ⊖woman-n, ⊖public-j, ⊕slave-n, ⊕present-v, ⊕detain-v, ⊕connect-v
woman-n	⊖old-n, ⊖reasonable-j, ⊖implant-n, ⊖black-n, ⊖man-n, ⊕integration-n, ⊕planet-n, ⊕focus-v, ⊕acceptance-n, ⊕use-n
system-n	⊖investment-n, ⊖ledger-n, ⊖interpretation-n, ⊖ductal-j, ⊖bus-n, ⊕assurance-n, ⊕da-n, ⊕variable-n, ⊕multiprocessor-n, ⊕aquifer-n
group-n	⊖whale-n, ⊖nominal-j, ⊖genetic-j, ⊖soil-n, ⊖correlation-n, ⊕supervision-n, ⊕substance-n, ⊕trained-j, ⊕delayed-j, ⊕exposure-n
child-n	⊖transfer-n, ⊖diabetic-j, ⊖temperament-n, ⊖marriage-n, ⊖creative-j, ⊕implant-n, ⊕underachieve-v, ⊕injury-n, ⊕difficulty-n, ⊕rape-n
life-n	⊖tribal-j, ⊖answer-n, ⊖style-n, ⊖element-n, ⊖change-n, ⊕relationship-n, ⊕coach-n, ⊕youth-n, ⊕battery-n, ⊕nomadic-j
student-n	⊖reverse-j, ⊖handicap-n, ⊖treatment-n, ⊖retarded-j, ⊖nondisabled-j, ⊕voucher-n, ⊕undocumented-j, ⊕nurse-v, ⊕algebra-n, ⊕bully-v
program-n	⊖supervisory-j, ⊖timber-n, ⊖thinking-n, ⊖tune-v, ⊖export-n, ⊕bully-v, ⊕yard-n, ⊕trimming-n, ⊕meditation-n, ⊕rating-n
work-n	⊖comparative-j, ⊖record-v, ⊖video-j, ⊖art-n, ⊖wage-n, ⊕curriculum-n, ⊕profession-n, ⊕travel-v, ⊕theory-n, ⊕literature-n
state-n	⊖downwind-j, ⊖store-n, ⊖terror-n, ⊖red-j, ⊖density-n, ⊕confidence-n, ⊕space-n, ⊕upwind-j, ⊕failed-j, ⊕pool-n
school-n	⊖victimization-n, ⊖field-n, ⊖samba-n, ⊖bureaucracy-n, ⊖entry-n, ⊕bully-v, ⊕bully-v, ⊕intervention-n, ⊕liaison-n, ⊕characteristic-n
way-n	⊖awkward-j, ⊖sensible-j, ⊖base-n, ⊖people-n, ⊕fun-n, ⊕point-n, ⊕big-j, ⊕food-n, ⊕indigenous-j
number-n	⊖seal-n, ⊖fisherman-n, ⊖stock-n, ⊖share-n, ⊖black-n, ⊕accounting-n, ⊕hometown-n, ⊕skill-n, ⊕cumulative-j, ⊕cell-n
day-n	⊖set-n, ⊖strike-n, ⊖government-n, ⊖quote-n, ⊖collection-n, ⊕adult-j, ⊕step-n, ⊕lust-n, ⊕smoke-v, ⊕be-v
area-n	⊖nursery-n, ⊖writer-n, ⊖cooperation-n, ⊖goal-n, ⊖development-n, ⊕story-n, ⊕wellhead-n, ⊕thought-n, ⊕necrosis-n, ⊕receive-v
level-n	⊖ethic-n, ⊖signal-n, ⊖fandom-n, ⊖cognition-n, ⊖output-n, ⊕identification-n, ⊕creatinine-n, ⊕lipid-n, ⊕parent-n, ⊕achievement-n

Table 3. Most trending adjectives in COCA computed by grouping word sketch triples.

more-j	⊖stock-n, ⊖hill-n, ⊖reward-n, ⊖belief-n, ⊕insulin-n, ⊕aggression-n, ⊕transparency-n, ⊕fantasy-n, ⊕green-n
such-j	⊖industry-n, ⊖special-j, ⊖good-n, ⊖stock-n, ⊖conduct-n, ⊕websites-n, ⊕library-n, ⊕meat-n, ⊕flavor-n, ⊕nurse-n
other-j	⊖saurs-n, ⊖match-n, ⊖fishery-n, ⊖payer-n, ⊕troll-n, ⊕covariates-n, ⊕websites-n, ⊕scandal-n, ⊕node-n
new-j	⊖atheist-n, ⊖packed-j, ⊖commonwealth-n, ⊖landfill-n, ⊖powder-n, ⊕physicist-n, ⊕normal-n, ⊕website-n, ⊕subscription-n, ⊕normal-j
good-j	⊖catfish-n, ⊖crappie-n, ⊖opponent-n, ⊖dealer-n, ⊕competition-n, ⊕teammate-n, ⊕bass-n, ⊕climber-n, ⊕learning-n
many-j	⊖skier-n, ⊖peasant-n, ⊖composer-n, ⊖debate-n, ⊖ingredient-n, ⊕intervention-n, ⊕sample-n, ⊕trend-n, ⊕challenge-n, ⊕challenge-n
first-j	⊖black-n, ⊖offender-n, ⊖complaint-n, ⊖instant-n, ⊕branchial-j, ⊕variate-n, ⊕responder-n, ⊕purpose-n, ⊕putt-n
own-j	⊖aide-n, ⊖superiority-n, ⊖prediction-n, ⊖century-n, ⊖psyche-n, ⊕drug-n, ⊕bar-n, ⊕take-n, ⊕online-j, ⊕wine-n
small-j	⊖orchard-n, ⊖stock-n, ⊖have-v, ⊖investor-n, ⊕channel-n, ⊕reactor-n, ⊕frequent-j, ⊕robot-n, ⊕buck-n
great-j	⊖bulk-n, ⊖dependence-n, ⊖heap-n, ⊖famine-n, ⊖loss-n, ⊕recession-n, ⊕face-n, ⊕mom-n, ⊕option-n, ⊕cast-n
little-j	⊖wing-n, ⊖dragon-n, ⊖look-n, ⊖argument-n, ⊕shabby-j, ⊕speak-v, ⊕camera-n, ⊕close-j, ⊕egg-n
large-j	⊖republic-n, ⊖laboratory-n, ⊖customer-n, ⊖lobster-n, ⊖hotel-n, ⊕heat-n, ⊕raw-j, ⊕lemon-n, ⊕nonstick-j, ⊕oven-n
high-j	⊖subject-n, ⊖scenario-n, ⊖rent-n, ⊖stage-n, ⊖gifted-j, ⊕youth-n, ⊕odd-n, ⊕dive-n, ⊕definition-n, ⊕career-n
old-j	⊖dance-n, ⊖economic-j, ⊖communist-j, ⊖viewer-n, ⊕year-n, ⊕participant-n, ⊕report-n, ⊕cartoon-n, ⊕dumb-j
big-j	⊖cigar-n, ⊖holding-n, ⊖look-n, ⊖newspaper-n, ⊖gamble-n, ⊕physical-j, ⊕celebrity-n, ⊕presence-n, ⊕opening-n, ⊕army-n
few-j	⊖stock-n, ⊖supporter-n, ⊖black-n, ⊖department-n, ⊕tweak-n, ⊕click-n, ⊕emission-n, ⊕string-n, ⊕bone-n
most-j	⊖price-n, ⊖skier-n, ⊖operator-n, ⊖tribe-n, ⊖therapist-n, ⊕component-n, ⊕phone-n, ⊕participant-n, ⊕liquid-n, ⊕county-n
same-j	⊖nation-n, ⊖tendency-n, ⊖shoe-n, ⊖department-n, ⊖restriction-n, ⊕interview-n, ⊕skillet-n, ⊕scrutiny-n, ⊕target-n, ⊕academic-j
different-j	⊖filter-n, ⊖relation-n, ⊖society-n, ⊖movement-n, ⊖political-j, ⊕landscape-n, ⊕stuff-n, ⊕night-n, ⊕demographic-j, ⊕rate-n

Table 4. Most trending verbs in COCA computed by grouping word sketch triples.

be-v	⊖logos-n, ⊖pellagra-n, ⊖shamanism-n, ⊖digester-n, ⊕ingrethents-n, ⊕authence-n, ⊕texting-v, ⊕diat-n, ⊕hed-n
have-v	⊖anthrax-n, ⊖mage-n, ⊖chickenpox-n, ⊕ames-n, ⊕blog-n, ⊕website-n, ⊕antidepressant-n, ⊕memorial-n
make-v	⊖newspaper-n, ⊖subject-n, ⊖district-n, ⊖antibody-n, ⊕sidebar-n, ⊕resilient-j, ⊕minute-n, ⊕tweak-n, ⊕backswing-n
do-v	⊖opponent-n, ⊖trail-n, ⊖aid-n, ⊖black-n, ⊕sidebar-n, ⊕adolescent-n, ⊕librarian-n, ⊕diligence-n, ⊕golfer-n
use-v	⊖order-v, ⊖ecstasy-n, ⊖fastball-n, ⊖airline-n, ⊕ingrethents-n, ⊕braille-n, ⊕metric-n, ⊕cellphones-n, ⊕website-n
get-v	⊖rocket-n, ⊖position-n, ⊖welfare-n, ⊖hostage-n, ⊕sidebar-n, ⊕text-n, ⊕stop-n, ⊕overwhelmed-j, ⊕com-n
go-v	⊖com-n, ⊖pond-n, ⊖station-n, ⊕viral-j, ⊕tobazaar-n, ⊕prevention-n, ⊕voice-v, ⊕sfgate-n
take-v	⊖leadership-n, ⊖overdose-n, ⊖capsule-n, ⊖paper-n, ⊕statins-n, ⊕meds-n, ⊕phone-n, ⊕sidebar-n, ⊕assessment-n
say-v	⊖corpse-n, ⊖republic-n, ⊖jazz-n, ⊖ing-n, ⊕website-n, ⊕demon-n, ⊕release-n, ⊕utility-n, ⊕down-x
come-v	⊖opposition-n, ⊖occupy-v, ⊖tell-v, ⊖program-n, ⊕kitchen-n, ⊕ghost-n, ⊕loss-n, ⊕online-j, ⊕remark-n
see-v	⊖earning-n, ⊖contract-n, ⊖doll-n, ⊖section-n, ⊕hardcopy-n, ⊕website-n, ⊕reference-n, ⊕online-j, ⊕http-n
become-v	⊖expression-n, ⊖cliche-n, ⊖card-n, ⊖officer-n, ⊖scholar-n, ⊕submission-n, ⊕guy-n, ⊕brand-n, ⊕cell-n, ⊕distraction-n
include-v	⊖survivor-n, ⊖wife-n, ⊖stretch-n, ⊖son-n, ⊖fare-n, ⊕covariates-n, ⊕band-n, ⊕determine-v, ⊕win-n, ⊕awareness-n
find-v	⊖bass-n, ⊖comet-n, ⊖residue-n, ⊖variety-n, ⊕sidebar-n, ⊕lesion-n, ⊕online-j, ⊕online-j, ⊕improve-v
begin-v	⊖tale-n, ⊖strike-n, ⊖black-n, ⊖therefore-a, ⊖listen-v, ⊕split-v, ⊕edge-n, ⊕archaeologist-n, ⊕arrive-v, ⊕dough-n
look-v	⊖age-n, ⊖hank-n, ⊖growth-n, ⊖court-n, ⊕amazing-j, ⊕jack-n, ⊕pair-n, ⊕photo-n, ⊕character-n
give-v	⊖black-n, ⊖resistance-n, ⊖judgment-n, ⊖urgency-n, ⊕how-x, ⊕prevalence-n, ⊕foot-n, ⊕certificate-n, ⊕scientist-n
seem-v	⊖spring-v, ⊖agency-n, ⊖century-n, ⊖dubious-j, ⊕solution-n, ⊕fun-n, ⊕die-v, ⊕set-v, ⊕smart-j
work-v	⊖piece-n, ⊖location-n, ⊖deal-n, ⊖medium-n, ⊖class-n, ⊕move-n, ⊕participant-n, ⊕batch-n, ⊕capacity-n, ⊕fluid-n
need-v	⊖telescope-n, ⊖finance-v, ⊖athlete-n, ⊖defense-n, ⊖satisfy-v, ⊕practitioner-n, ⊕nurse-n, ⊕miracle-n, ⊕option-n, ⊕dig-v

Table 5. Most trending other words in COCA computed by grouping word sketch triples.

his-d	⊖phaser-n, ⊖deer-n, ⊖quilt-n, ⊖envelope-n, ⊕stunner-n, ⊕blog-n, ⊕falcon-n, ⊕website-n, ⊕email-n
their-d	⊖blogs-n, ⊖playoff-n, ⊖national-n, ⊖banker-n, ⊕websites-n, ⊕cellphones-n, ⊕episode-n, ⊕primary-n, ⊕website-n
her-d	⊖administration-n, ⊖prejudice-n, ⊖cross-n, ⊖abortion-n, ⊕blog-n, ⊕fianc-n, ⊕rsum-n, ⊕website-n, ⊕reticule-n
its-d	⊖quarry-n, ⊖prospectus-n, ⊖recession-n, ⊖printing-n, ⊕website-n, ⊕footprint-n, ⊕heft-n, ⊕electricity-n, ⊕win-n
my-d	⊖stepmom-n, ⊖ski-n, ⊖cellphone-n, ⊖speculation-n, ⊕integrator-n, ⊕blog-n, ⊕participant-n, ⊕website-n, ⊕fianc-n
your-d	⊖infant-n, ⊖wreath-n, ⊖ski-n, ⊖skate-n, ⊕smartphone-n, ⊕protagonist-n, ⊕gadget-n, ⊕aspect-n, ⊕downswing-n
not-a	⊖transcribe-v, ⊖inconsistent-j, ⊖persuasive-j, ⊖wise-j, ⊖recur-v, ⊕access-v, ⊕post-v, ⊕manipulate-v, ⊕predetermine-v, ⊕assess-v
our-d	⊖mama-n, ⊖century-n, ⊖intimacy-n, ⊖cave-n, ⊕website-n, ⊕forum-n, ⊕specialist-n, ⊕shoot-n, ⊕blog-n
also-a	⊖smile-v, ⊖judge-v, ⊖guarantee-v, ⊖concede-v, ⊖desire-v, ⊕additional-j, ⊕detail-v, ⊕inject-v, ⊕picture-v, ⊕s-v
so-a	⊖label-v, ⊖sanguine-j, ⊖gracious-j, ⊖destructive-j, ⊖broad-j, ⊕freak-v, ⊕base-v, ⊕improbable-j, ⊕worth-j, ⊕welcome-v
then-a	⊖react-v, ⊖mail-v, ⊖negotiate-v, ⊖record-v, ⊖plunge-v, ⊕chill-v, ⊕top-j, ⊕halt-v, ⊕post-v, ⊕remind-v
as-a	⊖cry-v, ⊖blank-j, ⊖radical-j, ⊖give-v, ⊖dependent-j, ⊕delicious-j, ⊕test-v, ⊕die-v, ⊕place-v, ⊕connect-v
only-a	⊖attach-v, ⊖award-v, ⊖alternative-j, ⊖legal-j, ⊖token-j, ⊕straight-j, ⊕embody-v, ⊕clean-v, ⊕organic-j, ⊕motivate-v
still-a	⊖fume-v, ⊖uncertain-j, ⊖viable-j, ⊖evoke-v, ⊖suspect-v, ⊕link-v, ⊕relevant-j, ⊕sport-v, ⊕throw-v, ⊕crave-v
just-a	⊖flat-j, ⊖issue-v, ⊖eliminate-v, ⊖cite-v, ⊖wan-v, ⊕combine-v, ⊕wilt-v, ⊕shy-v, ⊕focus-v, ⊕tap-v
even-a	⊖strict-j, ⊖strike-v, ⊖such-j, ⊖contradictory-j, ⊖deadly-j, ⊕tear-v, ⊕accelerate-v, ⊕result-v, ⊕happy-j, ⊕par-j
very-a	⊖frighten-v, ⊖rapid-j, ⊖restrictive-j, ⊖definite-j, ⊖minor-j, ⊕persistent-j, ⊕tender-j, ⊕crowded-j, ⊕thankful-j, ⊕experience-v
now-a	⊖own-j, ⊖handle-v, ⊖concentrate-v, ⊖yield-v, ⊖pend-v, ⊕block-v, ⊕avoid-v, ⊕date-v, ⊕host-v, ⊕easy-j
often-a	⊖succeed-v, ⊖less-j, ⊖disappear-v, ⊖force-v, ⊖similar-j, ⊕frame-v, ⊕opt-v, ⊕whisk-v, ⊕host-v, ⊕stir-v
never-a	⊖permit-v, ⊖begin-v, ⊖practice-v, ⊖succeed-v, ⊖press-v, ⊕shy-v, ⊕write-v, ⊕recommend-v, ⊕underestimate-v, ⊕late-j

Converting the Corpus Query Language to the Natural Language

Daniela Ryšavá[1], Nikol Volková[1], Adam Rambousek[2]

[1] Faculty of Arts, Masaryk University
Arne Nováka 1, 602 00 Brno, Czech Republic
399364@mail.muni.cz, 399909@mail.muni.cz
[2] Natural Language Processing Centre,
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
rambousek@fi.muni.cz

Abstract. This paper presents the first version of a web application designed to convert queries in the Corpus Query Language to the natural language. The purpose of this application is to help users who want to learn and work with Corpus Query Language, providing tool to find out the explanation of the CQL query. In the future, when the application supports more features of CQL, it may be included as a hint in the web interface of a corpus search engine.

Keywords: Corpus Query Language, CQL, natural language generation, corpus

1 Introduction to Corpus Query Language

Corpus Query Language (CQL) [1,2] is a formal language that allows you to search corpus for grammatically complex patterns. This language was developed at the University of Stuttgart in 1990. Each part of the CQL query consists of a selected attribute and required value, enclosed in square brackets, in the form of `[attribute="value"]`. Apart from the whole word, it is possible to specify regular expressions as values, using special symbols and wildcards to substitute various character composition.

Most common are three types of attributes:

- **lemma**, basic word form, e. g. an infinitive for a verb, a nominative singular for a noun,
- **word**, word in the particular form,
- **tag**, morphological tag, e. g. Czech tags used by the majka tagger [3] *k1* for nouns, *c1* for nominatives, *nS* for singular.

Query may be modified with the exclamation mark to negate the value query, see Example 1. Users may expand the CQL query with various advanced features, some of them are explained in Tables 1 and 2.

Example 1. Query: `[word!="dog"]`
Interpretation: searched expression is not the word *dog*

Table 1. Operators

.	full stop	any character
[]	square brackets	any token
containing	structure containing	structure containing the searched expression
within	within structure	searched expression within structure

<s/>
<p/>
<doc/>
	structure = sentence
	structure = paragraph
	structure = document
		structure markup used with operators *containing* or *within*

Example 2. Query: `[lemma="ye.."]`
Interpretation: searched expression is a lemma beginning with *ye* followed by two characters: *year, yelk, yeti...*

Example 3. Query: `<s/> containing ([word="dog"][][lemma="cat"])`
Interpretation: searched expression is a sentence containing the word *dog* followed by one arbitrary token and the word *cat*

Example 4. Query: `([lemma="dog"][][lemma="cat"]) within <doc/>`
Interpretation: searched expression is the word *dog* followed by one arbitrary token and the word *cat* within a document

Table 2. Quantifiers

*	asterisk	iteration in range from zero to infinite
+	plus	iteration in range from one to infinite
?	question mark	iteration in range from zero to one
{n}	range in braces	exactly n iterations
{n,}	range in braces	iteration in range from n to infinite
{n,m} or {n, m}	range in braces	iteration in range from n to m

Example 5. Query: `[lemma="moo*"]`
Interpretation: searched expression is the word *mo, moo, mooo...* (the lemma *mo* with arbitrary iteration of the character *o*)

Example 6. Query: `[tag="k5.*"]+`
Interpretation: searched expression is the verb occurring at least one time

Example 7. Query: `[lemma="hoo?d"]`
Interpretation: searched expression is the lemma *hod* and *hood*

Example 8. Query: `[lemma="dog"] []{2} [lemma="cat"]`
Interpretation: searched expression is the lemma *dog* and *cat* with exactly two arbitrary tokens between them

Example 9. Query: `[word="po{2,}r"]`
Interpretation: searched expression is the word *poor* with two or more characters *o*

Example 10. Query: `[tag="k3.*"] []{0,3} [tag="k1.*"]`
Interpretation: searched expression is a pronoun and a noun with no more tokens between them, or up to three arbitrary tokens between them

2 Converting the query

The application is developed in Python 2 programming language. There are two ways to run the application, either as the command line script, or the web application. Both interfaces use the same back-end algorithm to generate the sentence in natural language. As of now, the application produces Czech sentences, but it is possible to add support for other languages.

Input CQL query is split to tokens (enclosed in square brackets), while also detecting any structure operators. In the next step, each token is processed separately, producing parts of the Czech sentence, taking into account both the token query, and any quantifiers. Afterwards, the complete sentence is generated.

The application supports only some features of the CQL and the query has to follow the standard CQL structure. The attribute of the query needs to be specified by typing *word, lemma* or *tag* and the value (the searched item) has to be enclosed in double quotation marks, e. g. *"dog"* or *"k1.*nP.*"*. For example, the query may follow the form `[lemma="dog"]` or `[tag="k1.*nP.*"]`.

2.1 Searching for words and lemmas

If users want to search for certain word or lemma, they use the default attribute-values queries, as seen in Figure 1.

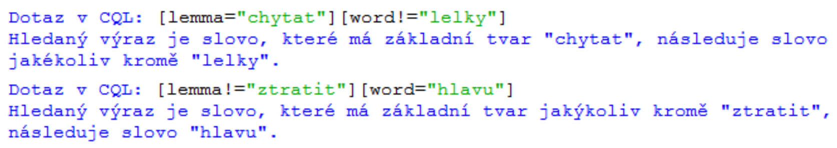
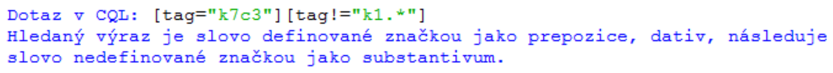

```
Dotaz v CQL: [lemma="chytat"][word!="lelky"]
Hledaný výraz je slovo, které má základní tvar "chytat", následuje slovo
jakékoliv kromě "lelky".
Dotaz v CQL: [lemma!="ztratit"][word="hlavu"]
Hledaný výraz je slovo, které má základní tvar jakýkoliv kromě "ztratit",
následuje slovo "hlavu".
```

Fig. 1. Searching for words and lemmas.

2.2 Searching for Part of Speech

Users may also search for particular morphological tags. In current version, the application supports the attributive tagset, used in the ajka/majka morphological analyzer[3]. If users want to specify combination of morphological tags, they have to enter the tags in the same order that is produced by the morphological analyzer. See Figure 2 for an example.

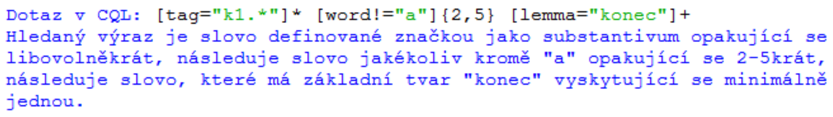


```
Dotaz v CQL: [tag="k7c3"][tag!="k1.*"]
Hledaný výraz je slovo definované značkou jako prepozice, dativ, následuje
slovo nedefinované značkou jako substantivum.
```

Fig. 2. Searching for POS.

2.3 Specifying number of tokens

It is possible to set the number of repeats for any token in the query, e. g. which parts of the query are required or optional. Number of tokens can be specified by quantifiers, i. e. ***, *?*, *+*, *{n}*, *{n,}*, *{n,m}* (see Table 2 for the list of quantifiers, and Figure 3 for an example).



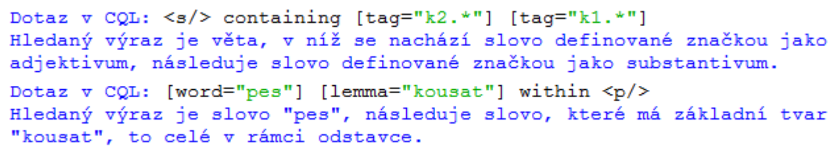
```
Dotaz v CQL: [tag="k1.*"]* [word!="a"]{2,5} [lemma="konec"]+
Hledaný výraz je slovo definované značkou jako substantivum opakující se
libovolněkrát, následuje slovo jakékoliv kromě "a" opakující se 2-5krát,
následuje slovo, které má základní tvar "konec" vyskytující se minimálně
jednou.
```

Fig. 3. Searching number of tokens.

2.4 Searching in structures

If the corpus has sentence, paragraph or document markup, it is possible to take these into consideration when writing CQL queries. Operators *s*, *p* and *doc* are valid structures and users are able to match tokens in such structures. The first type of the structure is specified with operator *containing*, and matches all structures of your choice containing e. g. an adjective followed by noun. The other type is specified with operator *within*, and matches a number of tokens within a structure of your choice. See Figure 4 for examples.

[3] For a complete list of tags, see `http://nlp.fi.muni.cz/projekty/ajka/tags.pdf`

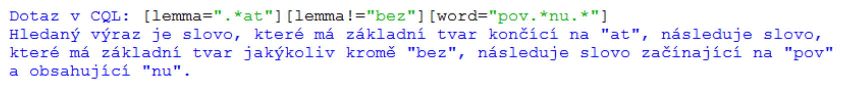


```
Dotaz v CQL: <s/> containing [tag="k2.*"] [tag="k1.*"]
Hledaný výraz je věta, v níž se nachází slovo definované značkou jako
adjektivum, následuje slovo definované značkou jako substantivum.
Dotaz v CQL: [word="pes"] [lemma="kousat"] within <p/>
Hledaný výraz je slovo "pes", následuje slovo, které má základní tvar
"kousat", to celé v rámci odstavce.
```

Fig. 4. Searching in structures.

2.5 Searching for complex patterns

If the value of a query includes any characters, other than letters or numbers, the query is evaluated as a regular expression. However, the current version of the application does not evaluate the value of a regular expression, because it is a complex tasks by itself. On the other hand, some frequent query types are detected. The application is able to detect the pattern .* in the CQL value, and from the position of this pattern in the query string decides whether the users are searching for words/lemmas starting with, ending with, or containing particular characters.



```
Dotaz v CQL: [lemma=".*at"][lemma!="bez"][word="pov.*nu.*"]
Hledaný výraz je slovo, které má základní tvar končící na "at", následuje slovo,
které má základní tvar jakýkoliv kromě "bez", následuje slovo začínající na "pov"
a obsahující "nu".
```

Fig. 5. Searching for complex patterns.

3 Conclusion and future work

The application is able to convert several frequent types of CQL queries to an explanation in Czech, even more advanced features like quantifiers or structure queries. Web application is available for users at `http://nlp.fi.muni.cz/projekty/cql2cz/`.

However, the application does not support all of the advanced CQL features, and current version has some issues with complex queries. We plan to address the issues and extend the CQL support, in future versions. In the next version, following areas will be covered.

3.1 The sequence of morphological categories

When users specify morphological tags with regular expressions, like *"k1.*c4"*, only the tags (*k1* and *c4*) are detected, and the rest of the query is stripped. However, the full regular expression may hold important information, thus need to be examined by the application more closely.

3.2 Regular expressions inside value

Currently, only the pattern .* is detected in the regular expressions. In future versions, the application will be able to detect other types of regular expressions correctly.

3.3 Complex patterns

The application should be able to detect and process more complex query structure, for example nested queries like `[word="haunt" and tag="k1.*"]`.

Acknowledgement This work has been partly supported by the Masaryk University within the project *Čeština v jednotě synchronie a diachronie – 2015* (MUNI/A/1165/2014) and by the Ministry of Education of CR within the LINDAT-Clarin project LM2010013.

References

1. Evert, S.: The CQP query language tutorial. (2005)
2. Sketch Engine, Corpus Querying: `https://www.sketchengine.co.uk/xdocumentation/wiki/SkE/CorpusQuerying`
3. Jakubíček, M., Kovář, V., and Šmerk, P.: Czech Morphological Tagset Revisited. In *Proceedings of Recent Advances in Slavonic Natural Language Processing* (2011) 29–42

Concurrent Processing of Text Corpus Queries

Radoslav Rábara and Pavel Rychlý

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

Abstract. The fast evaluation of complex queries on big text corpora is an important feature of corpus managers. The aim of this paper is to apply approaches of concurrent processing to the query evaluation in the corpus management system Manatee. The work contains an evaluation of the query processing speed using various number of cores available, and also a comparison of the length of the source code between the original and the concurrent implementation.

Keywords: query evaluation, concurrency, Manatee, Go, lines of code

1 Introduction

Text corpora are huge collections of texts in electronic form. It is used as a resource of the empirical language data, i.e. words, their meanings and contexts they occur in. Corpora can be employed in many fields of linguistics (morphology, syntax, semantics, stylistics, sociolinguistics etc.) and the corpus managers are primary tools enabling corpus exploration. The corpus managers has to be able to deal with extremely large corpora and to provide a platform for evaluating complex queries, filtering and visualizing results, and computing a wide range of lexical statistics. The speed is an important aspect of the operations because the corpora are usually large – up to billions of words.

In order to speed up the evaluation of the operations, we reimplemented a single-thread evaluation with the concurrent approach within the Manatee corpus management system [5]. The reimplemented system, referred to as the new implementation, was compared to the original system. Not only the performance was evaluated and compared, but also the number of lines of source code was compared.

2 Manatee system

The Manatee system [5] is a corpus manager, it is able to deal with extremely large corpora and the important aspect is the speed of the queries evaluation.

The program has modular design. There is an indexing library for compression, building and retrieving indexes; a query evaluation module with classes for different query operations, a query parser which transforms the queries into abstract syntactic trees, a set of command line tools for corpus building and maintenance, two graphical user interfaces.

The system is based on the text indexing library FinLib that provides procedures for word indexing, corpus storage and retrieving words in form of streams of positions [4]. Manatee has its own query language, which enables retrieving of the word's occurrences.

2.1 FastStream and RangeStream

The implementation of a query evaluation within Manatee is based on streams of tokens or token pairs (representing range of consecutive tokens). The FastStream and RangeStream interfaces (abstract classes) represent token and range streams. Classes which implements them, represent specific operations. The main idea here is to have classes that perform simple operations which can be combined together to perform complex operations.

In the original implementation, these classes are based on the iterator pattern. It means that there is always available only one value from the stream of values. The next value is loaded by calling the `next` method. Once it is called, the previous values are no longer available. Values are always provided in increasing order. After all values are read, an iterator returns a stopper, which is an arbitrary value greater than any value in the stream. Iterator also provides the `find` method to lookup efficiently in the remaining values and the `peek` method to get the following value, which will be returned by calling the `next` method, without proceeding to the next value. More details about the implementation are in [4].

3 The Go programming language

Go, also referred to as golang[1], is a new programming language which has been developed since 2007 as Google project. Go tries to combine performance and security advantages of compiled language like C++ with the development speed of a dynamic language like Python [2]. It was developed by famous programmers such as Ken Thompson[2], author of the Unix operating system, Rob Pike[3], also a member of the Unix developing team, or Robert Griesemer[4], who worked on the design and implementation of the Java HotSpot virtual machine.

The concurrent running functions are called goroutines in Go. It is a coroutine attached to a thread. Multiple coroutines can be attached to a single thread. The attachment is performed dynamically by the Go runtime. The

[1] `https://golang.org/`
[2] `https://en.wikipedia.org/wiki/Ken_Thompson`
[3] `https://en.wikipedia.org/wiki/Rob_Pike`
[4] `https://en.wikipedia.org/wiki/Robert_Griesemer`

attachment of the coroutine can be changed to another operating system thread when a different coroutine attached to the same thread is blocked, e.g. by calling a blocking system call.

Communication between and synchronization of the concurrent running goroutines can be provided by channels. Channels exchange data from the sender to receiver. The communication blocks the sender until the receiver receives the data. In this way, the goroutines provides synchronization of goroutines without explicit locks or condition variables.

Channels can be buffered. The buffered channels blocks the sender only when the buffer is full and it blocks the receiver only when the buffer is empty. More details about Go channels could be found in [3] and many tutorials at the Go site: `golang.org`.

The new implementation was written in Go.

4 Implementation

In the new implementation, FastStream and RangeStream are structures that provide channel as the stream of the positions. The original methods `next`, `peek`, and `find` are no longer provided because the stream of the positions is represented by the channel. Also the operation `find` was removed from the new implementation because the operation was slowing down the application almost in all cases, although it was expected that the operation improves the application performance by reducing the amount of the transmitted data.

FastStream and RangeStream were renamed as FastChan and RangeChan in the new implementation. This naming convention, which uses suffix “Chan” instead of “Stream”, emphasizes data transfer over the channels.

In the original implementation, FastStream and RangeStream are interfaces and the classes which implements them, represent specific operations. The new implementation does not have classes implementing FastStream and RangeStream interfaces, but there are functions with FastChan or RangeChan as the return value. The functions create channel, run a goroutine, and return the created channel. The goroutine executes computation and the results are send via the channel, which is closed after the computation is done. This approach is generally known as the generator pattern [1].

4.1 Data exchange between goroutines

An interesting paradox was discovered during testing of the initial concurrent design. The faster computation was expected when the program was running on more than a single core, but the performance was worse. It seemed not only that the computation is not running in parallel, but also that the program is slowed down by runtime task scheduler. Measured results are showed in Figure 1.

The program with the initial concurrent design contained only three goroutines. The program performed the operation of intersection (operation AND)

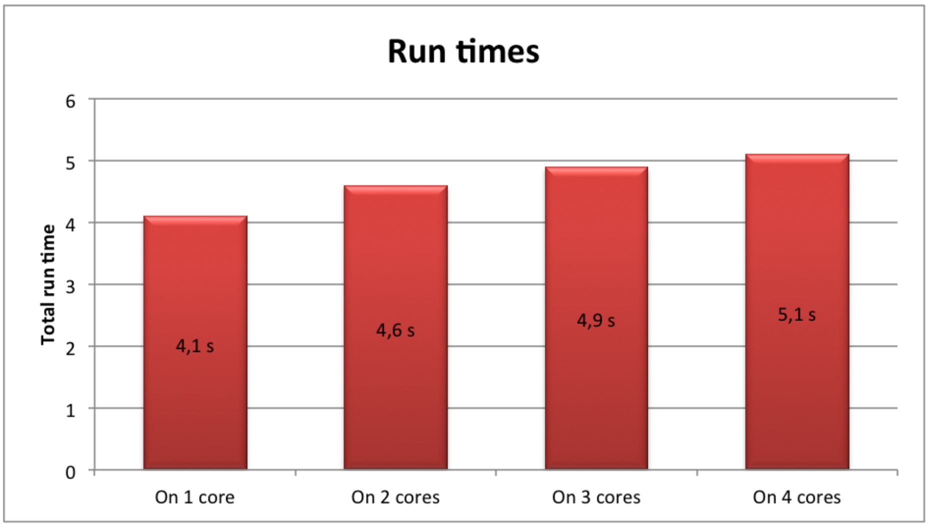


Fig. 1. More cores cause worse performance.

on two channels —- streams of the positions. The streams were loaded from the hard disc in separate goroutines. The relatively slow operation of reading from hard disc could run in parallel with the computation of the operation AND.

The operation AND does not execute time consuming instructions between receiving the data from the input channels. It consists from the simple loop which receives positions from the both inputs and compares the positions. If the positions are equal, it sends the position to the output and loads the next positions from both streams. If the positions are not equal, it loads next position from stream that contains lower position.

Channels use locks to archive synchronous communication and the used design repeately sends and receives data after performing relatively fast computation so the program was slowed down on multiple cores probably because of constantly accessing the channel from multiple cores. Therefore, we tried to send batches containing multiple positions instead of sending single positions.

Sending multiple positions at once causes less usage of channels. Goroutine receives batch of positions, processes the whole batch, and then receives the another batch. The larger the batch is, the bigger delays are between receiving batches from the channel. Disadvantage is waiting for the first batch, which must be filled by a sending goroutine. Advantage can be preloading of batch into the cache memory while adding to or reading positions from the batch.

The results of the program which transmits the batches of the positions instead of the single positions, are presented in Figure 2. The application performs the same operations but the performance is positively influenced by the number of used cores.

In Figure 2, we see that the program speeded up when it run on two cores comparing to the run on one core. It is a proof that data transmission was the issue of the scalability. We can also see that running the program on more than two cores does not cause significant improvement of the performance. It is related to the simplicity of the tested program.

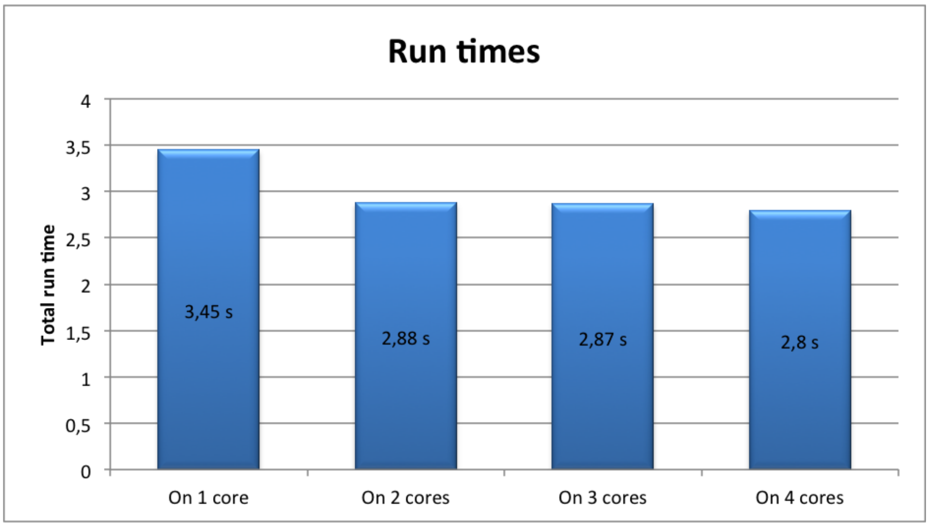


Fig. 2. Sending batch of positions improves performance.

4.2 Comparing the performance

The evaluation of the both implementations was performed on eight processor core server. The original implementation is a single-thread application, so it can use only one processor core. The new implementation has the concurrent design with goroutines, which can run on multiple cores. The new implementation was evaluated with the various number of the cores.

The performance was compared by a simple benchmark that measured time of the evaluation of the prepared queries. The time was measured by the unix `time` command. The prepared evaluation queries are quite difficult and complex because they cover rules of syntactic analysis. The results of these queries are quite big as they cover from 5 to 10 % of the whole corpus. Together the results cover almost the whole corpus. These queries are quite extreme because typical users of the Manatee system usually create just simple queries to find some specified words with some restrictions, which produce much smaller results.

The original implementation evaluated the prepared queries in 4h 29m 24s. The average of the total run (real) times of the queries evaluation are presented in Figure 3.

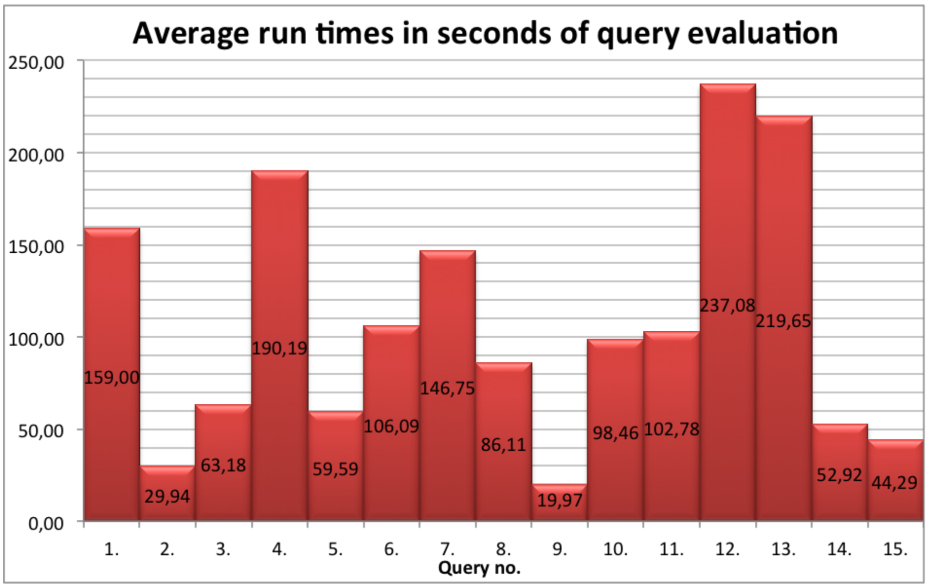


Fig. 3. The average run times of query evaluation by the original implementation.

The new implementation evaluated the evaluation queries in 2h 27m 39s, when it run only with one core. Comparing to the original implementation, the new implementation is faster by approximately 45 %. The avarage of the total run (real) times of queries evaluation are presented in Figure 4.

The Figure shows that thirteen from fifteen evaluation queries were evaluated faster by the new implementation with an average difference of approximatelly 45 %. The fourteenth query had the best improvement by 82 %. The second query had the smallest improvement by 20 %. Two queries were evaluated slower by the new implementation: the ninth query, which was evaluated slower by 116 %, and the eleventh query, which was evaluated slower by 7 %.

The new implementation was evaluated with all possible number of processor cores on the same server, so we can see how the performance is affected when the new implementation is executed on more cores. Measurements are displayed in Figure 5.

The most significant difference is between running on one and two cores, which speeds up the evaluation by 43 %. When the new implementation runs on more than two cores, the performance is better with the each added core, but the cores addition results in slightly less difference of the performance. Only

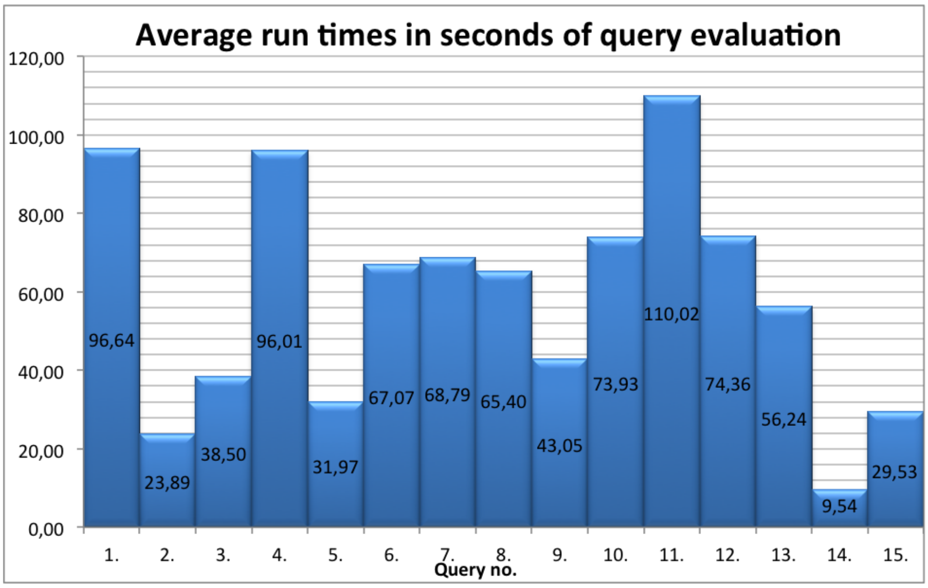


Fig. 4. The average run times of query evaluation by the new implementation.

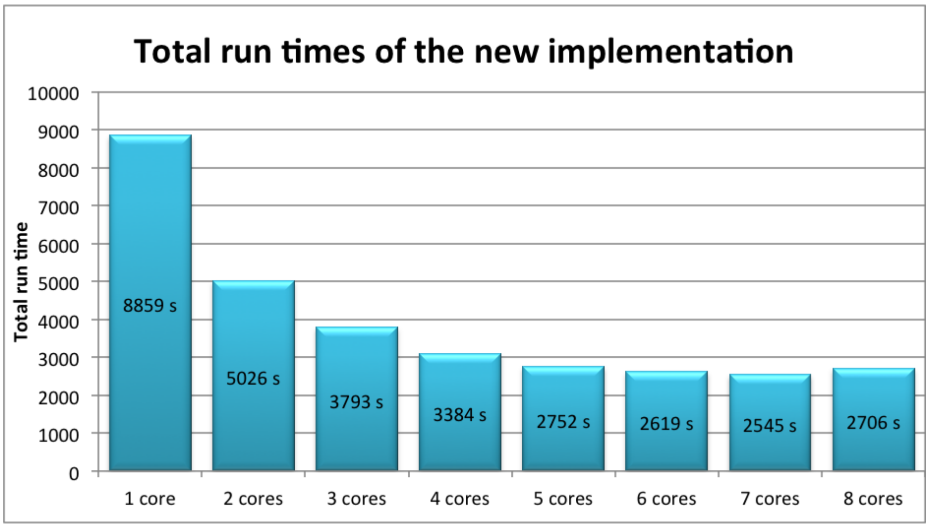


Fig. 5. The total run time of the new implementation running on all possible number of processor cores.

when the new implementation run with the all eight cores the performance was worse than running with only seven or six cores.

The evaluation with three cores speeds up the evaluation by 24 % comparing to the evaluation with two cores and by 57 % comparing to the evaluation with one core. Four cores speeds up the evaluation by 18 % comparing to the evaluation with three cores and by 65 % comparing to the evaluation with one core. Adding the fifth core increases the performance by 11 % comparing to the evaluation with four cores. Runs with six and seven core differ in less than 5 % comparing to the runs without one core.

4.3 Comparing the lines of code

The source code of the original implementation is written in C++, which separates the source code to header files, which contain declaration of an interfaces, and source files, which contain implementation of the declared interfaces. The header files can declare classes, functions or variables. It can also contain implementations of short functions. Working with the source code requires reading from the both — header and source files. Therefore, both file types are included in the source code analysis of the original implementation.

The source code of the new implementation is written in Go, which does not have more types of source code files and it does not require linking between files in the same module. However, it has strict syntactic rules that can prolong the source code.

The source code of the new implementation is expanded by the implemented mechanism of the batch transmission, which provides scalable data transmission used to evaluate queries. The C++ has advantage of the initialization lists[5] which shorts attribute initialization to one line.

The strict syntactic rules in Go forbids one line functions and writing if statement without surrounding block. Go also forbids some common expressions in C++ such as `variable1 = variable2++;`.

The one line functions has its header (return type, function name and parameters) and body (implementation) on the same line. Functions in Go consume at least three lines: the first one consists of a function header and the start of the function's body, the second one contains statement, and the third one is the end of the function's body.

The new implementation does not have all functionality of the original system and therefore, the comparision cannot include the whole source code. Instead, only some units were compared. The units can be divided into two groups:

1. FastStream (FastChan), RangeStream (RangeChan), and structures which provides batch transmisison. The difference between lines of codes indicates how the length of source code was affected by changing from a single-thread to concurrency paradigm.
2. SortedRuns, Read_bits, and Write_bits, which have the same implementation so the difference between lines of codes indicates how the length of

[5] `http://www.cprogramming.com/tutorial/initialization-lists-c++.html`

source code was affected by changing the programming language from C++ to Go.

The results include only the not blank and not comment lines, and only those units of the code were included, which does not provide functionality unimplemented in the new system. The results are presented in Figure 6.

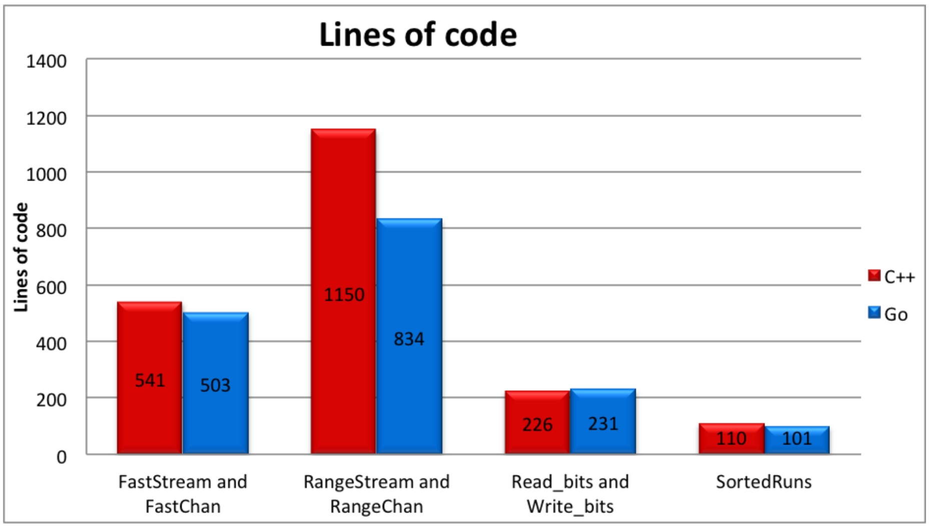


Fig. 6. Graph comparing lines of code of the original and the new implementation.

The number of lines of the reimplemented FastStream and RangeStream interfaces and classes is reduced by 21 %, so the change of the sequential to concurrent paradigm leaded to the shorter source code. The difference could be even bigger if there would not be needed mechanisms of sending and receiving positions as batches, which takes 252 lines of code (19 %).

Reimplementing of Read_bits and Write_bits slightly expanded the source code and even that the reimplemented SortedRuns has less code by 1 %, the reimplemented code of the second group has rather longer code.

5 Conclusion

The performance and the length of the source code were compared between the single-thread and concurrent implementation of the corpus manager Manatee. Manatee is able to deal with extremely large corpora and to provide a platform for evaluating complex queries, filtering and visualizing results, and computing a wide range of lexical statistics [5].

The original implementation (in C++) evaluated the prepared benchmark queries in aproximatelly 4.5 hours. The new implementation (in Go) evaluated the prepared benchmark queries on one core in approximatelly 2.5 hours. The concurrent system has performance better by 45 % on one core. The new implementation was also evaluated with all possible combinations of processor cores. The performance is enhanced by 15.7 % in average by each core of the server and the most significant enhancement by 43 % was between running on the one and two cores.

The new implementation does not have all functionality of the original system so the comparision of the length of code was processed only with some units of the code. The source code analysis points out that disadvantage of C++ are header files and disadvantages of Go are strict syntactic rules and the implementation of the data structures that deal with batch data transmission. The original implementation has 2027 lines of code of the selected units and the new implementation has 1667 lines of code of the appropriate units.

Evaluations proof that the new concurrent implementation has better performance and shorter source code. Therefore, it will replace the original system after the missing functionality will be implemented.

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

References

1. Dolan, R.: Go Language Patterns: Generators. (2009) Available from: `https://sites.google.com/site/gopatterns/concurrency/generators`.
2. Kincaid, J.: Google's Go: A New Programming Language That's Python Meets C++. (2009) Available from: `http://techcrunch.com/2009/11/10/google-go-language/`.
3. Pike, R.: Concurrency is not parallelism. Heroku's Waza conference (2012). `http://talks.golang.org/2012/waza.slide`.
4. Rychlý, P.: Corpus Managers and their effective implementation. PhD Thesis, Faculty of Informatics, Masaryk University (2000)
5. Rychlý, P.: Manatee/Bonito - A Modular Corpus Manager In 1st Workshop on Recent Advances in Slavonic Natural Language Processing. Masaryk University, Brno (2007) 65–70

Part III

Lexicons and Information Extraction

Bilingual Terminology Extraction in Sketch Engine

Vít Baisa[1,2], Barbora Ulipová, Michal Cukr[1,2]

[1] Natural Language Processing Centre,
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
baisa@fi.muni.cz
[2] Lexical Computing
Brighton, United Kingdom and Brno, Czech Republic
{vit.baisa,michal.cukr}@sketchengine.co.uk

Abstract. We present a method of bilingual terminology extraction from parallel corpora and a few heuristics and experiments with improving the performance of the basic variant of the method. An evaluation is given using a small gold standard manually prepared for English-Czech language pair from DGT translation memory [1]. The bilingual terminology extraction (ABTE[3]) is available for several languages in Sketch Engine—the corpus management tool [2].

Keywords: terminology extraction, bilingual terminology extraction, Sketch Engine, logDice, parallel corpus

1 Introduction

Parallel corpora are valuable resources for machine and computer-assisted translation. Here we explore a possibility of extracting bilingual terminology from parallel corpora combining a monolingual terminology extraction [3] and a co-occurrence statistics [4]. We describe the method and how it is incorporated in the corpus manager tool Sketch Engine. We experimented with parameter tuning and evaluated a few settings using a small gold standard for English-Czech language pair.

The following section is a brief survey of topics, methods and tools in ABTE. In Section 3 we describe the basic algorithm for the extraction and in Section 4 how it is integrated in Sketch Engine. In Sections 5 and 6 we evaluate the algorithm and its variants.

2 Related work

The monolingual terminology extraction is a well-studied field, and the topic of ABTE has been explored since 90s [5] but recent summarizing publication [6]

[3] ATE stands for "automatic terminology extraction", so we adopt the abbreviation here and add "B" for "bilingual".

do not mention this area of research: it is not yet a well-established area of the terminology field. It is probably partially caused by the fact that the quality of ABTE methods is rather poor.

ABTE functions are available in several commercial tools, e.g. MultiTerm[4], Araya Bilingual Extraction Tool[5], but it is hard to find any particular numbers referring to the quality of these tools.

Several ABTE-related papers can be split into two groups: 1) those using a combination of a monolingual extraction together with a co-occurrence statistics (e.g. [5], our algorithm belongs here too) and 2) those implementing an alternative approach.

An example of an alternative approach is described in [7] where authors used a bootstrapping technique: they started with discovering confident terminology pairs and then extracted rules for correspondence between terms in different languages (e.g. French-Dutch rule N+ADJ $\leftrightarrow$ ADJ+N).

Another approach is to use the phrase-based translation model approach as authors of [8]. According to [7], multi word:single word terms ratio is 7:3[6] so it is not viable to resort to single word alignment methods. It is though possible to start with a word alignment (using e.g. the Expectation-maximization algorithm) and then to extract pairs of phrases which are consistent with the word alignment.

3 The algorithm

Technically, parallel corpora in Sketch Engine are stored as monolingual corpora. The parallel alignment (usually on the sentence level) is defined by a mapping of IDs of special `<align>` structures or IDs of sentences (alternatively paragraphs and documents). The former is used when a 1:1 alignment is available (e.g. when working with TMX files, which is the case for DGT translation memory [1]). The latter is used in all other cases (e.g. in OPUS2 [9], EuroParl [10]).

The process of extraction of parallel terms consists of several steps. The first step is the monolingual terminology extraction in both languages. The procedure is described in [3]. The important thing to mention here is that all term candidates are found by means of matching grammar rules defined with CQL[7] (corpus query language) which are usually matching noun phrases in various forms.[8] For the ABTE method described here, it is not necessary to sort the (monolingual) term candidates by termhood by comparing term candidate frequency in a focus corpus with a reference corpus [3].

In a next step, the algorithm computes co-occurrence statistics for all aligned structures and for all candidate pairs occurring within the aligned structures.

[4] `www.sdl.com/cxc/language/terminology-management/multiterm`

[5] `www.heartsome.de/en/termextraction.php`

[6] It was measured in the domain of automotive industry.

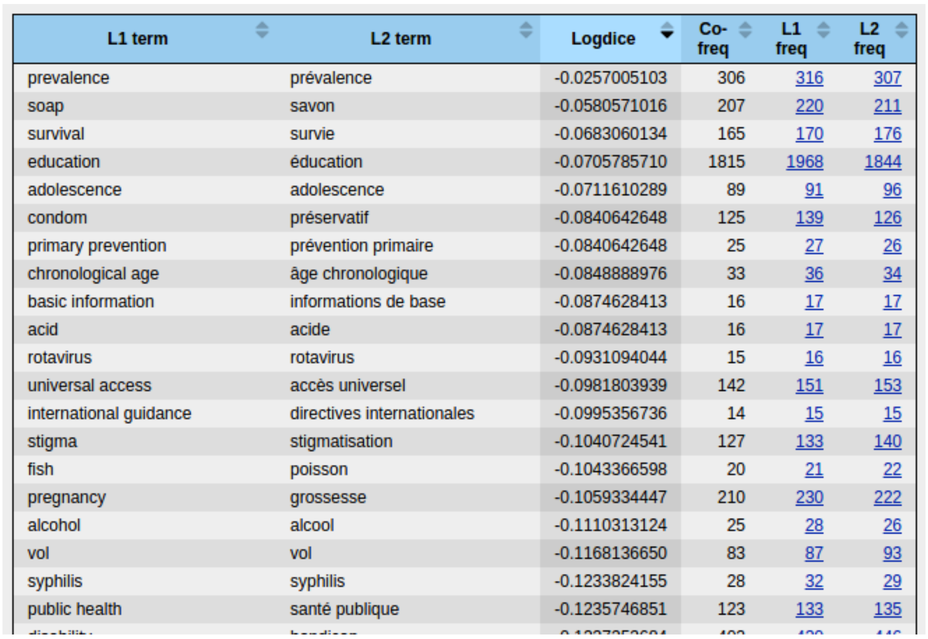
[7] `www.sketchengine.co.uk/corpus-querying`

[8] According to [7], the majority of terms is in the form of noun phrases.

The resulting list of all term pairs can be sorted by various scores. By default Sketch Engine uses logDice[9] [**?**].

4 Bilingual terminology extraction in Sketch Engine

Bilingual terminology extraction is one part of a complex Sketch Engine's pipeline for building parallel corpora. For ABTE to work, it is necessary to have the monolingual terminology extracted for all languages in the relevant language pairs. Supported languages are: Chinese, Czech, Dutch, English, French, German, Italian, Japanese, Korean, Polish, Portuguese, Russian, Spanish.



L1 term	L2 term	Logdice	Co-freq	L1 freq	L2 freq
prevalence	prévalence	-0.0257005103	306	316	307
soap	savon	-0.0580571016	207	220	211
survival	survie	-0.0683060134	165	170	176
education	éducation	-0.0705785710	1815	1968	1844
adolescence	adolescence	-0.0711610289	89	91	96
condom	préservatif	-0.0840642648	125	139	126
primary prevention	prévention primaire	-0.0840642648	25	27	26
chronological age	âge chronologique	-0.0848888976	33	36	34
basic information	informations de base	-0.0874628413	16	17	17
acid	acide	-0.0874628413	16	17	17
rotavirus	rotavirus	-0.0931094044	15	16	16
universal access	accès universel	-0.0981803939	142	151	153
international guidance	directives internationales	-0.0995356736	14	15	15
stigma	stigmatisation	-0.1040724541	127	133	140
fish	poisson	-0.1043366598	20	21	22
pregnancy	grossesse	-0.1059334447	210	230	222
alcohol	alcool	-0.1110313124	25	28	26
vol	vol	-0.1168136650	83	87	93
syphilis	syphilis	-0.1233824155	28	32	29
public health	santé publique	-0.1235746851	123	133	135

Fig. 1. The bilingual terminology extraction in Sketch Engine. The default sort criterion is logDice but the list of candidates can be sorted also by the co-occurrence frequency. The frequency numbers are links to monolingual concordances.

5 Gold standard and experiments

First, we tried to get existing gold standard data. We asked authors of two papers [7,11] for their gold standard data used for the evaluation but they

[9] `www.sketchengine.co.uk/wp-content/uploads/ske-stat.pdf`

could not provide us with it due to possible copyright issues. Another option was to use an official terminology base from the European Union institution Directorate-General for Translation IATE[10] (Interactive Terminology for Europe) and run the extraction on a translation memory from the same institution, DGT translation memory [1]. This is also not possible as the IATE is not fully compatible with DGT translation memory.

That is why we have prepared our own gold standard for English-Czech pair. We manually cleaned 1,000 term pairs from a run of our algorithm, optimized for a high-coverage output. We did not take into account whether the items in the list were actually terms or not as we didn't want to evaluate the monolingual terminology extraction. We only decided whether the terms were translated correctly and whether they covered the same scope of the term. This means that for example the translation of the term "dry linen content" – "obsah suchého prádla" was considered as correct while the translation "suchého prádla" of "dry linen" was considered incorrect. This particular error was caused by the monolingual terminology extraction step as it is an incorrect gender-dependent base form. The correct form "suché prádlo" was not found in this case. We removed 66% of the 1,000 term pairs.

The annotated terms were then divided into two files—a file with correctly translated terms (the gold standard) containing 328 term pairs and a file with incorrectly translated terms. The gold standard file then used to evaluate the output of different settings of the algorithm. We used standard metrics precision, recall, and F-1 score. For ABTE, recall is usually more important than precision as users expect to post-edit and clean up candidate lists but do not want to miss possible term candidates. Therefore, the modifications we used were designed to increase precision and at the same time not decreasing recall.

The modifications were: 1) preferred terms with fewer words in the first language L1, 2) different ratios of the number of characters in L1 and L2 and 3) discarding the terms with low co-occurrence.

The terms with fewer words were preferred by the following method. The formula uses two variables: the number of words in L1 and a coefficient by how much the shorter terms should be preferred. Smaller coefficient prefers shorter terms.

$$\frac{4 - \text{number of words}}{10 * \text{coefficient}} + 1$$

In the process of sorting the final list we multiplied logDice score numerator by the formula above. The second modification uses one variable ratio, which we call *ideal ratio*, and which states the ideal ratio between the length of words or phrases in L1 and L2. For example, the ideal ratio 0.8 means that the pairs in L1 which have 20% fewer characters than in L2 will be preferred. First, the real ratio, i.e. the ratio between the number of characters in the L1 and L2 is counted.

[10] `iate.europa.eu`

$$\text{real ratio} = \frac{\text{characters in L1}}{\text{characters in the L2}}$$

If the real ratio is smaller than the ideal ratio, we count the mod:

$$\text{mod} = \frac{\text{real ratio}}{\text{ideal ratio}}$$

If the real ratio is bigger than ideal ratio or equal, we count the mod:

$$\text{mod} = \frac{\text{ideal ratio}}{\text{real ratio}}$$

We then multiply logDice with the mod.

In the last modification, we discarded pairs with co-occurrence lower than 4. The gold standard does not contain any terms with co-occurrence under 4. Therefore, we did not decrease recall.

The gold data set is available for download[11] under CC BY-SA licence[12].

6 Evaluation

We tested the algorithm on the gold standard. While we tried different ratios, we found out that ratios 0.92–0.97 give the best results. This means that English terms should be slightly shorter than the Czech terms. It corresponds with the fact that English texts are approximately 1.1 times shorter than Czech.[13]

We also tested the preference of the terms having fewer words in the first language. The range of the coefficient which we tested was 1–6. However, all the settings where the coefficient was higher than 1 (the terms with fewer words had less preference) were significantly less successful. Furthermore, we tried to discard and to not discard the terms with low co-occurrence frequency. The results were better when the terms with low co-occurrence frequency were discarded. This was expected as mentioned above. All the settings discard the terms with low co-occurrence because those settings always yielded better results. The ratio column contains the ideal ratio. The first line contains original settings where the ratio and the shorter terms are not preferred.

The Table 1 shows that the best results were achieved with ratio between 0.92–0.97. We obtained 1.24 times better results with the experiments against the original settings.

7 Conclusions

We described an approach to ABTE already implemented in corpus management tool Sketch Engine. We also evaluated the algorithm using a manually

[11] `www.sketchengine.co.uk/bilingual-terminology-extraction`

[12] `creativecommons.org/licenses/by-sa/2.0`

[13] `www.eurotranslation.cz/index.php?menu=faq`

Table 1. Evaluation, the best results.

RATIO	PRECISION	RECALL	F-SCORE
N/A	0.3282	0.3232	0.3257
2.00	0.3994	0.3933	0.3963
0.98–1.5	0.4025	0.3964	0.3994
0.92–0.97	0.4056	0.3994	**0.4025**
0.91	0.4025	0.3964	0.3994
0.90	0.3994	0.3933	0.3963
0.80	0.3993	0.3933	0.3963
0.75	0.3963	0.3902	0.3932

prepared gold standard for English-Czech language pair from DGT translation memory and suggested a possible improvement of the method.

In the future we would like to try an alternative approach to ABTE in the form of an on-the-fly searching for translation candidates from parallel concordances for arbitrary phrases during a computer-assisted translation process.

The ABTE methods in general are not supposed to completely eliminate the need of a painstaking manual post-editing of terminology bases. The candidate lists are intended as a starting point for building bilingual terminology bases from scratch which can significantly speed up the process of otherwise tedious and time-consuming work.

Acknowledgement This work has been partly supported by the Masaryk University within the project *Čeština v jednotě synchronie a diachronie – 2015* (MUNI/A/1165/2014).

References

1. Steinberger, R., Andreas, E., Szymon, K., Spyridon, P., Patrick, S.: Dgt-tm: A freely available translation memory in 22 languages. Proceedings of the 8th international conference on Language Resources and Evaluation (LREC'2012) (2012)
2. 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
3. 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
4. Rychlý, P.: A lexicographer-friendly association score. Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN (2008) 6–9
5. Kupiec, J.: An algorithm for finding noun phrase correspondences in bilingual corpora. In: Proceedings of the 31st annual meeting on Association for Computational Linguistics, Association for Computational Linguistics (1993) 17–22
6. Kockaert, H.J., Steurs, F.: Handbook of Terminology. Volume 1. John Benjamins Publishing Company (2015)
7. Macken, L., Lefever, E., Hoste, V.: Texsis: Bilingual terminology extraction from parallel corpora using chunk-based alignment. Terminology **19**(1) (2013) 1–30

8. Itagaki, M., Aikawa, T., He, X.: Automatic validation of terminology translation consistency with statistical method. Proceedings of MT summit XI (2007) 269–274
9. Tiedemann, J.: News from opus-a collection of multilingual parallel corpora with tools and interfaces. In: Recent advances in natural language processing. Volume 5. (2009) 237–248
10. Koehn, P.: Europarl: A parallel corpus for statistical machine translation. In: MT summit. Volume 5., Citeseer (2005) 79–86
11. Rösiger, I., Schäfer, J., George, T., Tannert, S., Heid, U., Dorna, M.: Extracting terms and their relations from german texts: Nlp tools for the preparation of raw material for specialized e-dictionaries. (2015)

Corpus Based Extraction of Hypernyms in Terminological Thesaurus for Land Surveying Domain

Vít Baisa, Vít Suchomel

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

Abstract. We present two methods of automatic hypernym extraction implemented within a dictionary editing and writing system for terminological thesaurus for land surveying domain: a specialised corpus based method and a term similarity approach. Challenges of both techniques are briefly discussed.

Keywords: thesaurus, terminological dictionary, corpus building, semantic relation, hypernym extraction, land surveying

1 Introduction

This work is an extension of dictionary editing and writing system for terminological thesaurus for land surveying domain [2]. The system consists of a terminology database and a text corpus. The terms are organized in a tree structure representing semantic relations hypernymy and hyponymy. The corpus is used for displaying the context of terms in domain-related documents (the concordance function), for finding synonyms[1] automatically (the thesaurus function) and extracting new terms (the term extraction function).

Having a terminology dictionary allowing terminologists to add their own documents to the underlying corpus and to extract new terms from the documents, a terminologist needs a function for placing the extracted terms into the existing tree structure of terms. In other words, the system has to provide a hypernym for every new term. Although there are methods for automatic hypernym finding (see below), the final decision must be made by a human expert. Therefore, it is useful to provide more hypernym candidates and let the terminologist decide which are true hypernyms or define the correct hypernym (or hypernyms) manually.

In this paper we present two methods for automatic hypernym extraction implemented within the terminological dictionary: a specialised corpus based method and a term similarity driven approach. Challenges of both techniques affecting the usability of hypernym candidates in the system are briefly discussed.

[1] More precisely, semantic similarity based on a shared collocational context.

2 Related Work

The task of finding semantic relations between terms, especially hypernymy, is very similar to the task of finding definitions for terms. Once a definition (definiens) of a term is available, it is likely to contain a hypernym or synonyms of the given term (definiendum). Therefore this work has been inspired by [4] whose authors deal with finding English definitions in large corpora. A corpus is used as a source of definition texts. It is queried for patterns of words typical for definitions. We have exploited the same idea for extraction of hypernyms of Czech terms.

[3] reports low precision and recall for Polish, so the method may perform worse in morphologically richer languages than English.

Patterns of co-occurrences of words in semantic relations in Czech were studied in [5] and [6]. We have chosen the best patterns according to evaluations of those studies.

3 Extending the Specialised Corpus

The specialised corpus for land surveying and geo-information domain built in an earlier stage of the project [2] has been used. To improve the extraction of semantic relations, cadastre, land surveying, and geo-information related Czech laws and regulations were downloaded from the respective web sites and added into the corpus. Because of their purpose, these documents are likely to contain term definitions and terms in hypernym relations.

The corpus was augmented by the following texts obtained from the web portal of State Administration of Land Surveying and Cadastre (`http://www.cuzk.cz`) in July 2015:

– Overview of law regulations concerning land surveying and cadastre (24 documents).[2]
– Law regulations in land surveying and cadastre (13 documents).[3]
– Department regulations and actions (43 documents).[4]
– Data sets (5 documents).[5]
– INSPIRE (Infrastructure for spatial information in European Union) (14 documents).[6]

The extended corpus has been compiled and indexed for fast search in corpus manager Manatee/Bonito [1]. The current total size of the corpus is 12,691,252 positions (9,757,005 words including 3,864,481 nouns) in 27,389 documents.

[2] `http://www.cuzk.cz/Predpisy/Prehled-pravnich-predpisu-souvisejicich-se-zememer.aspx`

[3] `http://www.cuzk.cz/Predpisy/Pravni-predpisy-v-oboru-zememerictvi-a-katastru.aspx`

[4] `http://www.cuzk.cz/Predpisy/Resortni-predpisy-a-opatreni/Pokyny-CUZK-1-15.aspx`, `http://www.cuzk.cz/Predpisy/Resortni-predpisy-a-opatreni/Pokyny-CUZK-16-30.aspx`, `http://www.cuzk.cz/Predpisy/Resortni-predpisy-a-opatreni/Pokyny-CUZK-31-42.aspx`

[5] `http://geoportal.cuzk.cz/%28S%28qn4tqoqg02oega1vqydc1o4g%29%29/Default.aspx?head_tab=sekce-02-gp&mode=TextMeta&text=dSady_uvod&menu=20&news=yes`

[6] `http://geoportal.cuzk.cz/%28S%28lyfis0b5dkkvrox2b5b1q2h2%29%29/Default.aspx?head_tab=sekce-04-gp&mode=TextMeta&text=inspire_uvod&menu=40&news=yes`

4 Hypernym Extraction Methods

Two methods of automatic hypernym extraction were implemented within the terminological dictionary: a specialised corpus driven method and a term similarity based approach.

4.1 Corpus Based Extraction

The specialised domain corpus used for various functions in the system can also be exploited for hypernym extraction. The corpus is queried through the concordance API of Sketch Engine [1] for the following patterns in the CQL formalism[7]. The extracted hypernym candidates are sorted by log-Dice score:

$$\text{similarity} = \log_2 \left(\frac{2 * \text{number of co-occurrences of both terms}}{\text{term 1 frequency} + \text{term 2 frequency}} \right)$$

Pattern 1: The hyponym + **is/are** *(je/jsou)* + the hypernym.[8]

```
2:[k="k1"&c="c1"] ([lc=","] [k="k1"])*([lc="a|i|nebo|či"] [k="k1"])?
[lemmalc="být"&tag="k5eAaImIp3.*"&lc!="ne.*"]
([k="k1"&c="c[1246]"] [k="k2"]{0,2})? 1:[k="k1"&c="c[1246]"] within <s/>
```

Examples of terms in hypernymic relation extracted by Pattern 1:

- loxodroma $\subset$ křivka
- teodolit $\subset$ geodetický přístroj
- územní řízení $\subset$ správní řízení

Pattern 2: The hyponym + **and/or another/other/similar** *(a/nebo další/jiný/ostatní/podobný)* + the hypernym:

```
2:[k="k1"] ([lc=",|a|nebo|či"] [k="k1"])* [lc="a|i|nebo|či|zejména|ani"]
[lemmalc="také|též|některý|nějaký|než"]?
[lemmalc="další|jiný|ostatní|podobný"]
([k="k1"&c="c[1246]"] [k="k2"]{0,2})? 1:[k="k1"&c="c1"] within <s/>
```

Examples of terms in hypernymic relation extracted by Pattern 2:

- elektronický teodolit $\subset$ měřický přístroj
- hospodářský pozemek $\subset$ pozemková držba
- mapový znak $\subset$ kartografický vyjadřovací prostředek

Pattern 3: The hyponym + **is/are kind/type/part/example/way of** *(je/jsou druhem/typem/částí/příkladem/způsobem)* + the hypernym:

[7] Corpus Querying Language. Developed at the Corpora and Lexicons group, IMS, University of Stuttgart. The CQL as used in Sketch Engine is an extension to the original language.

[8] The hypernym token is labelled by `1` and the hyponym token is labelled by `2` in all CQL queries here.

```
2:[k="k1"&c="c1"] ([lc=","] [k="k1"])* ([lc="a|i|nebo|či"] [k="k1"])?
[lemmalc="být"&tag="k5eAaImIp3.*"&lc!="ne.*"]
[k="k1"&(lemmalc="druh|typ|část|příklad|způsob")]
1:[k="k1"&c="c2"] within <s/>
```

Examples of terms in hypernymic relation extracted by Pattern 3:

– ionosféra ⊂ atmosféra
– Morfometrie ⊂ kartometrie
– pozemek ⊂ zemský povrch

4.2 Term Similarity Approach

The other implemented method of finding hypernyms of a term is searching the term database. The given term is compared to all existing terms in the system database. The most similar terms are expected to be good hypernym/hyponym candidates. We have adopted Jaccard distance of bigrams of characters with threshold of 0.5 as the similarity measure of two terms:

$$\text{similarity} = \frac{|\text{term 1 bigrams} \cap \text{term 2 bigrams}|}{|\text{term 1 bigrams} \cup \text{term 2 bigrams}|}$$

Examples of terms in hypernymic relation found in the database using the similarity measure:

– absolutní tíhový bod ⊂ tíhový bod
– fáze Měsíce ⊂ Měsíc
– způsob ochrany nemovitosti ⊂ nemovitost

Both methods are combined in the system and the best candidates for hypernyms are available for the terminologists in the user interface in the form of a select box. An example of the (Czech) user interface is given in Figure 1.

5 Evaluation and Discussion

5.1 Corpus Based Extraction

The evaluation of top 25 and top 50 pair candidates extracted from the specialised corpus and sorted by log-Dice is shown in Table 1.

Although the accuracy of Pattern 1 and Pattern 2 queries is above 50 %, not all successfully extracted hypernym pairs are suitable for the particular term database. For example, term 'mapové dílo' ('map series') is a hyponym of 'dílo' ('series') however term 'kartografické dílo' ('cartographic product') – that is already in the term database – is a much more suitable hypernym.

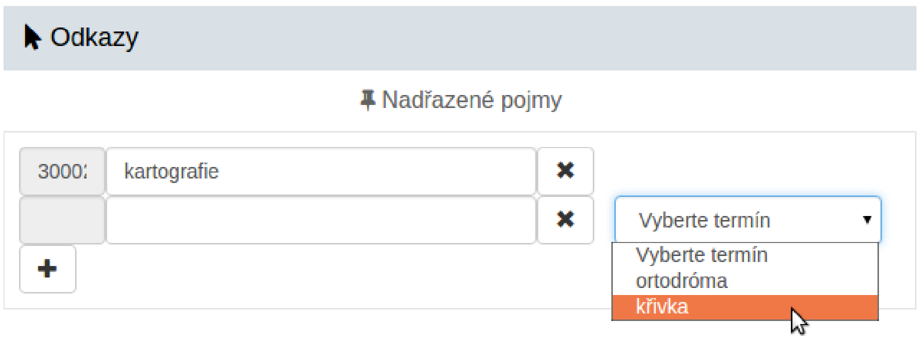


Fig. 1. Automatic hypernym suggestions as implemented in the system. A part of editing form for the term 'loxodróma' is shown. Two terms visible in the select box are automatically suggested hypernyms.

Table 1. Percentage of hypernym pairs in all candidate pairs extracted from the corpus.

Pattern 1 – **'is'**		Pattern 2 – **'and other'**		Pattern 3 – **'is kind of'**	
candidates	hypernyms	candidates	hypernyms	candidates	hypernyms
top 25	52 %	top 25	52 %	top 25	0 %
top 50	56 %	top 50	56 %		

5.2 Term Similarity Approach

50 random terms from the database having at least one hypernym candidate were evaluated. The best three hypernym candidates were taken in account (the same as in the application). A hypernym was identified among the three most similar candidates in 56 % of cases.

Again, some successfully extracted hypernym pairs might not be suitable for the particular term database, because of e.g. a level in the hypernym tree is skipped or added or there is a better hypernym which was not identified by the automatic method.

That is why the final decision which candidate to select or whether to input the hypernym manually is left to a human expert.

6 Conclusion

We have successfully extended a dictionary editing and writing system for terminological thesaurus for land surveying domain by automatic hypernym extraction. The new function helps terminologists to keep the tree structure of the term database organised by suggesting candidates for hypernyms of terms.

Acknowledgements This work has been partly supported by the Ministry of Education of CR within the LINDAT-Clarin project LM2010013 and 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**(1) (2014) 7–36.
2. Horák, A., Rambousek, A., Suchomel, V., Kocincová, L.: Semiautomatic Building and Extension of Terminological Thesaurus for Land Surveying Domain. In: Eighth Workshop on Recent Advances in Slavonic Natural Language Processing. Brno: Tribun EU, 2014. pp. 129–137.
3. Przepiorkowski, A., Degorski, L., Wojtowicz, 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, ACL 2007. pp. 43–50.
4. Kovář, V., Močiariková, M., Baisa, V., Jakubíček, M.: Finding Definitions in Large Corpora with Sketch Engine. To appear. (Submitted to LREC 2016.)
5. Uhlíř, P.: Automatické vyhledávání nadřazených a podřazených pojmů v textu. Bachelor's thesis. Masaryk university. `http://is.muni.cz/th/255461/fi_b/`.
6. Haken, P.: Automatická extrakce sémanticky příbuzných slov. Bachelor's thesis. Masaryk university. `http://is.muni.cz/th/139525/fi_b/`.

Software and Data for Corpus Pattern Analysis

Vít Baisa[1], Ismaïl El Maarouf[2], Pavel Rychlý[1], Adam Rambousek[1]

[1] Natural Language Processing Centre,
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
{xbaisa,pary,rambousek}@fi.muni.cz
[2] Oxford University Press,
Oxford, United Kigdom
elmaarouf.ismail@gmail.com

Abstract. This report describes the tools and resources developed to support Corpus Pattern Analysis (CPA)—a corpus-based method for building patterns dictionaries. The tools are an annotation of concordance in Sketch Engine, a special CPA editor for editing Pattern Dictionary of English Verbs (PDEV), dedicated servlets based on the Dictionary Editing and Browsing platform and a public interface for browsing the PDEV. The resources are SemEval 2015 Task 15 dataset and LEMON API.

Keywords: Corpus Pattern Analysis, Pattern Dictionary of English Verbs, Sketch Engine, linked open data, ontology, LEMON

1 Introduction

In this report we present the suite of tools and datasets developed to support the construction of the Pattern Dictionary of English Verbs (PDEV). PDEV is the main output of Corpus Pattern Analysis (CPA), a novel technique in corpus linguistics to map meaning of words onto their patterns of use as observed in real texts. Section 2 gives a brief overview of CPA, Sections 3, 4, 5, 6 present the tools and interfaces used by CPA lexicographers. Sections 7 and 8 introduce a recent work in using the lexicographical resources for NLP. The Bibliography section is a comprehensive compilation of all major publications related to CPA and PDEV[3] not necessarily related directly to this report.

2 Corpus Pattern Analysis

Corpus Pattern Analysis (CPA) is a procedure in corpus linguistics which associates word meaning with word use by means of analysis of phraseological

[3] And its Spanish and Italian counterparts developed by Irene Renau's team at Pontifical Catholic University of Valparaíso, Chile available online www.verbario.com and by Elisabetta Jezek's team at the University of Pavia, Italy, respectively.

patterns and collocations. In CPA, no attempt is made to identify the meaning of a verb or noun directly, as a word in isolation. Instead, meaning is associated with prototypical sentence contexts. Concordance lines are grouped into semantically motivated syntagmatic patterns. Associating meaning with each pattern is a secondary step, carried out in close coordination with the assignment of concordance lines to patterns. The identification of a syntagmatic pattern is not an automatic procedure: it calls for a great deal of lexicographic art. Among the most difficult of all lexicographic decisions is the selection of an appropriate level of generalization on the basis of which senses are to be distinguished. For example, one might say that the in-transitive verb *abate* has only one sense (to become less in intensity), or one might separate *a storm abates* from *a political protest abates*, on the grounds that the two contexts have different implicatures.[4]

A large apparatus of linguistic categories has been progressively developed to capture corpus patterns for verbs. Patterns can be described according to five types of arguments: Subject, Object, Complement, Adverbial, and Indirect Object. Each can be further specified using determiners, semantic types, contextual roles[5], and lexical sets. Determiners are used to account for distinctions between "take place" and "take his place". Semantic types account for distinctions such as "building [[Machines]]" and "building [[Relationship]]". Contextual roles account for distinctions such as "[[Human = Film Director]] shoot" and "[[Human = Sports Player]] shoot". Lexical sets account for distinctions such as "reap the whirlwind" and "reap the harvest".

PDEV is maintained with three main tools: Sketch Engine [1], the CPA editor and the DEB server. The corpus used is the BNC [2], a large reference corpus containing various text types in British English (100 million words). For the purpose of the CPA analysis, the corpus has been filtered and only its written part was used. The result is usually referred as BNC50 since it contains roughly 50 million words. Lexicographers extract typical phraseological patterns from corpora by clustering corpus tokens (labelling them) according to the similarity of their context.

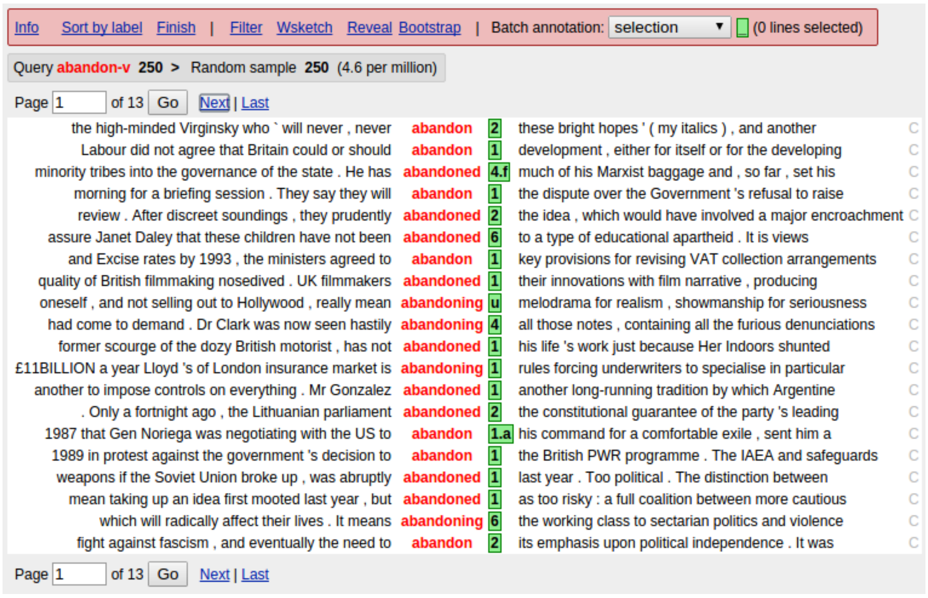
3 Annotation in Sketch Engine

Sketch Engine supports the annotation of tokens in any corpus using unique identifiers that refer to labels manually defined by lexicographers. The starting point is the creation of a concordance, based on a lemma and part-of-speech tag (lempos). In the web interface [3], the annotation is facilitated by green boxes next to each KWIC (Key Word In Context). The set of labels consists of numbers (for pattern numbers) although words can be used instead and each label can be further decomposed into sub-labels indicating a variation of use

[4] The implicature is a term from pragmatics referring to what is suggested in an utterance.

[5] Semantic types and roles are enclosed in double square brackets, roles are separated by "=".

('.a' for anomalous argument, '.f' for figurative use of a pattern, and '.s' for syntactically anomalous). Label 'x' is used for tagging errors (e.g. an adjective use for a verb token) and 'u' for an unclassifiable word occurrence. Moreover, a label can be assigned to a whole page, or to a set of selected lines. The set of labels can be modified at any time.


Info Sort by label Finish | Filter Wsketch Reveal Bootstrap | Batch annotation: selection (0 lines selected)

Query abandon-v 250 > Random sample 250 (4.6 per million)

Page 1 of 13 Go Next | Last

the high-minded Virginsky who ` will never , never	abandon	2	these bright hopes ' (my italics) , and another	C
Labour did not agree that Britain could or should	abandon	1	development , either for itself or for the developing	C
minority tribes into the governance of the state . He has	abandoned	4.f	much of his Marxist baggage and , so far , set his	C
morning for a briefing session . They say they will	abandon	1	the dispute over the Government 's refusal to raise	C
review . After discreet soundings , they prudently	abandoned	2	the idea , which would have involved a major encroachment	C
assure Janet Daley that these children have not been	abandoned	6	to a type of educational apartheid . It is views	C
and Excise rates by 1993 , the ministers agreed to	abandon	1	key provisions for revising VAT collection arrangements	C
quality of British filmmaking nosedived . UK filmmakers	abandoned	1	their innovations with film narrative , producing	C
oneself , and not selling out to Hollywood , really mean	abandoning	u	melodrama for realism , showmanship for seriousness	C
had come to demand . Dr Clark was now seen hastily	abandoning	4	all those notes , containing all the furious denunciations	C
former scourge of the dozy British motorist , has not	abandoned	1	his life 's work just because Her Indoors shunted	C
£11BILLION a year Lloyd 's of London insurance market is	abandoning	1	rules forcing underwriters to specialise in particular	C
another to impose controls on everything . Mr Gonzalez	abandoned	1	another long-running tradition by which Argentine	C
. Only a fortnight ago , the Lithuanian parliament	abandoned	2	the constitutional guarantee of the party 's leading	C
1987 that Gen Noriega was negotiating with the US to	abandon	1.a	his command for a comfortable exile , sent him a	C
1989 in protest against the government 's decision to	abandon	1	the British PWR programme . The IAEA and safeguards	C
weapons if the Soviet Union broke up , was abruptly	abandoned	1	last year . Too political . The distinction between	C
mean taking up an idea first mooted last year , but	abandoned	1	as too risky : a full coalition between more cautious	C
which will radically affect their lives . It means	abandoning	6	the working class to sectarian politics and violence	C
fight against fascism , and eventually the need to	abandon	2	its emphasis upon political independence . It was	C

Page 1 of 13 Go Next | Last

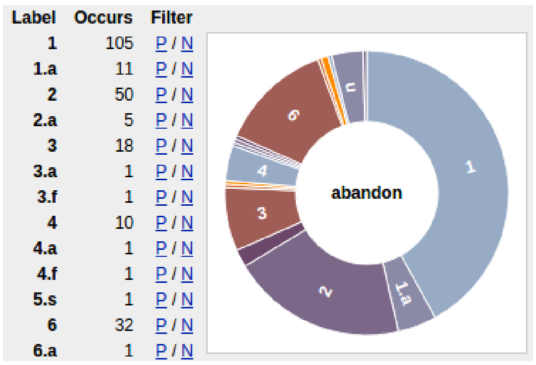
Fig. 1. The annotation interface in Sketch Engine.

Labels are stored as IDs internally which allows for renaming labels in batches. The mapping files are stored one per concordance. It is also possible to 1) undo an action, 2) sort the concordance according to the label values, 3) combine with standard sorting (by left or right context), 4) visualize the annotation statistics (Figure 2), 5) filter concordance by labels or 6) annotate via word sketches.

The labelling actions are the part of Bonito API[6]: each assignment of labels to line(s) is sent via AJAX to the server and saved. The API has methods for assigning labels, renumbering labels (changing the mapping globally for an annotation), locking the current annotation, obtaining annotation statistics etc.

A training mode is also available to enable trainees to annotate a concordance sample without altering the existing annotations. In this mode each annotation by a trainee is stored separately and can be viewed by a trainer. The performance of each trainee with respect to the master concordance can also be

[6] `www.sketchengine.co.uk/json-api-documentation`



Label	Occurs	Filter
1	105	P / N
1.a	11	P / N
2	50	P / N
2.a	5	P / N
3	18	P / N
3.a	1	P / N
3.f	1	P / N
4	10	P / N
4.a	1	P / N
4.f	1	P / N
5.s	1	P / N
6	32	P / N
6.a	1	P / N

Fig. 2. Visualization of annotation statistics for verb "abandon".

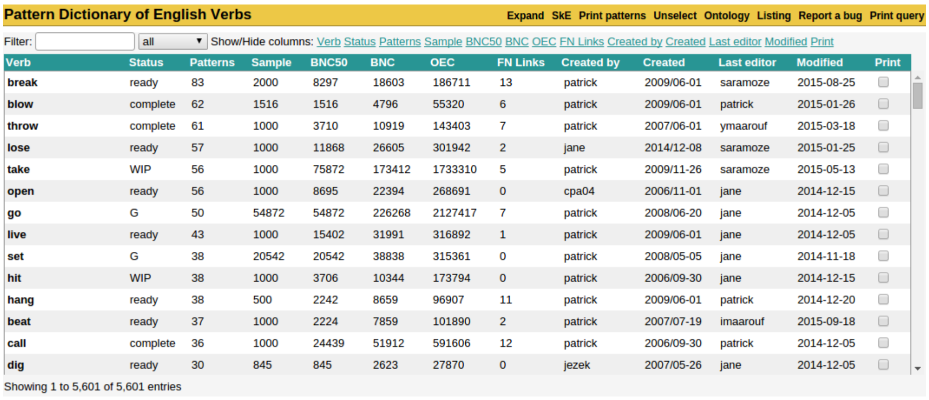
computed. Finally a trainee's concordance can be validated by the trainer and merged with the main concordance.

Other features have been developed such as an automatic clustering of concordance lines, a bootstrapping mechanism to annotate the rest of a partially annotated concordance using collocation-based features identified in the context of the lines already annotated. The evaluation of the bootstrap was given in [4].

4 CPA editor

The CPA editor is implemented in JavaScript (using jQuery and DataTables) and PHP. It mainly consists of an entry manager (Figure 3) and a pattern editor (Figure 4). The entry manager retrieves information about entries and stores them in a DataTable table, thus enabling easy sorting of columns and filtering using the search input box or verb statuses (complete, ready, work in progress). Each line of the table can be selected/unselected to be printed in CSV. When a user clicks on an entry, the entry manager opens an instance of a pattern editor in a new tab, which is the workspace for the entry.

The pattern editor is organized in three main parts. 1) The top bar contains information for the entry such as the sample size, the status or comments. 2) The pattern list displays patterns and their frequency in the annotated concordance, the pattern string, the implicature and also indications about whether the pattern is an idiom or whether it is used in a particular register. 3) Clicking on any of the pattern line opens a pattern box (Figure 5). This box lists arguments line by line and each line can be expanded to account for alternating components (such as the fact that two semantic types are equally valid in a given argument). Various information can be added in the form of check boxes, menu lists, etc. The final line is the implicature, which can be automatically generated based on the pattern string which adds up all the elements in the pattern.

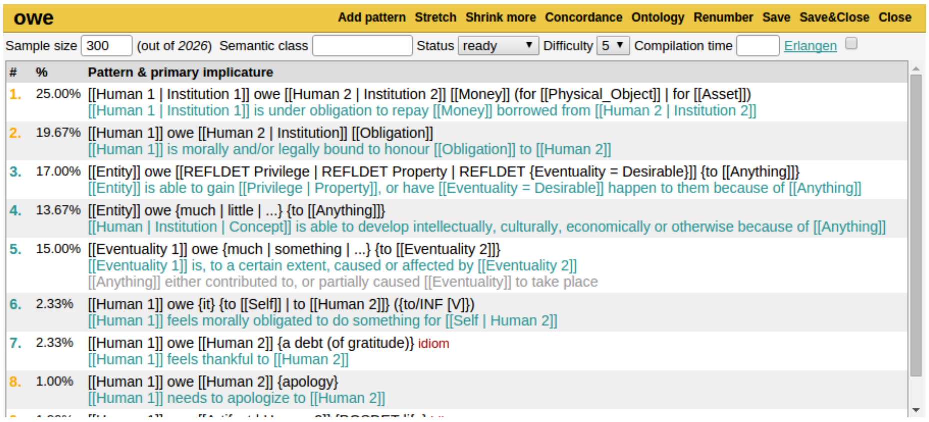

Pattern Dictionary of English Verbs — Expand SkE Print patterns Unselect Ontology Listing Report a bug Print query

Filter: all — Show/Hide columns: Verb Status Patterns Sample BNC50 BNC OEC FN Links Created by Created Last editor Modified Print

Verb	Status	Patterns	Sample	BNC50	BNC	OEC	FN Links	Created by	Created	Last editor	Modified	Print
break	ready	83	2000	8297	18603	186711	13	patrick	2009/06-01	saramoze	2015-08-25	☐
blow	complete	62	1516	1516	4796	55320	6	patrick	2009/06-01	patrick	2015-01-26	☐
throw	complete	61	1000	3710	10919	143403	7	patrick	2007/06-01	ymaarouf	2015-03-18	☐
lose	ready	57	1000	11868	26605	301942	2	jane	2014/12-08	saramoze	2015-01-25	☐
take	WIP	56	1000	75872	173412	1733310	5	patrick	2009/11-26	saramoze	2015-05-13	☐
open	ready	56	1000	8695	22394	268691	0	cpa04	2006/11-01	jane	2014-12-15	☐
go	G	50	54872	54872	226268	2127417	7	patrick	2008/06-20	jane	2014-12-05	☐
live	ready	43	1000	15402	31991	316892	1	patrick	2009/06-01	jane	2014-12-05	☐
set	G	38	20542	20542	38838	315361	0	patrick	2008/05-05	jane	2014-11-18	☐
hit	WIP	38	1000	3706	10344	173794	0	patrick	2006/09-30	jane	2014-12-15	☐
hang	ready	38	500	2242	8659	96907	11	patrick	2009/06-01	patrick	2014-12-20	☐
beat	ready	37	1000	2224	7859	101890	2	patrick	2007/07-19	imaarouf	2015-09-18	☐
call	complete	36	1000	24439	51912	591606	12	patrick	2006/09-30	patrick	2014-12-05	☐
dig	ready	30	845	845	2623	27870	0	jezek	2007/05-26	jane	2014-12-05	☐

Showing 1 to 5,601 of 5,601 entries

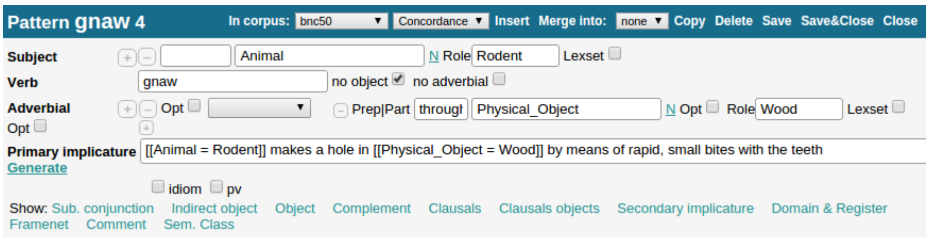
Fig. 3. CPA editor – the entry manager.


owe — Add pattern Stretch Shrink more Concordance Ontology Renumber Save Save&Close Close

Sample size 300 (out of *2026*) Semantic class Status ready Difficulty 5 Compilation time Erlangen ☐

#	%	Pattern & primary implicature
1.	25.00%	[[Human 1 | Institution 1]] owe [[Human 2 | Institution 2]] [[Money]] (for [[Physical_Object]] | for [[Asset]]) [[Human 1 | Institution 1]] is under obligation to repay [[Money]] borrowed from [[Human 2 | Institution 2]]
2.	19.67%	[[Human 1]] owe [[Human 2 | Institution]] [[Obligation]] [[Human 1]] is morally and/or legally bound to honour [[Obligation]] to [[Human 2]]
3.	17.00%	[[Entity]] owe [[REFLDET Privilege | REFLDET Property | REFLDET {Eventuality = Desirable}]] {to [[Anything]]} [[Entity]] is able to gain [[Privilege | Property]], or have [[Eventuality = Desirable]] happen to them because of [[Anything]]
4.	13.67%	[[Entity]] owe {much | little | ...} {to [[Anything]]} [[Human | Institution | Concept]] is able to develop intellectually, culturally, economically or otherwise because of [[Anything]]
5.	15.00%	[[Eventuality 1]] owe {much | something | ...} {to [[Eventuality 2]]} [[Eventuality 1]] is, to a certain extent, caused or affected by [[Eventuality 2]] [[Anything]] either contributed to, or partially caused [[Eventuality]] to take place
6.	2.33%	[[Human 1]] owe {it} {to [[Self]] | to [[Human 2]]} ({to/INF [V]}) [[Human 1]] feels morally obligated to do something for [[Self | Human 2]]
7.	2.33%	[[Human 1]] owe [[Human 2]] {a debt (of gratitude)} idiom [[Human 1]] feels thankful to [[Human 2]]
8.	1.00%	[[Human 1]] owe [[Human 2]] {apology} [[Human 1]] needs to apologize to [[Human 2]]

Fig. 4. CPA editor – the pattern manager.


Pattern gnaw 4 — In corpus: bnc50 — Concordance — Insert Merge into: none — Copy Delete Save Save&Close Close

Subject	Animal	N Role Rodent	Lexset ☐		
Verb	gnaw	no object ☑	no adverbial ☐		
Adverbial	Opt ☐	Prep|Part througl	Physical_Object	N Opt ☐	Role Wood Lexset ☐
Opt ☐					

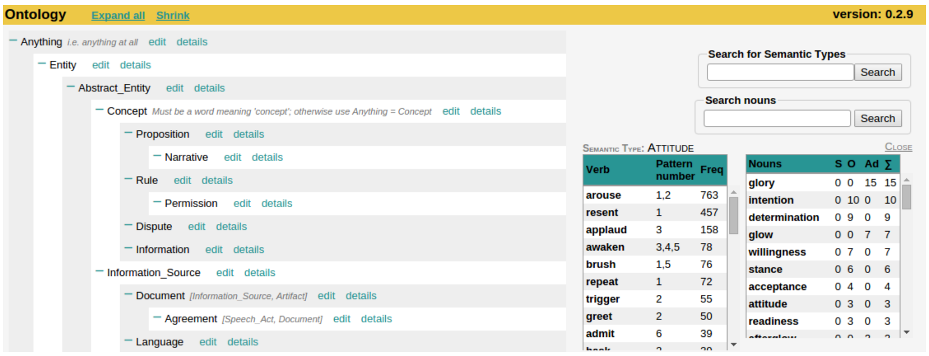
Primary implicature [[Animal = Rodent]] makes a hole in [[Physical_Object = Wood]] by means of rapid, small bites with the teeth
Generate

☐ idiom ☐ pv

Show: Sub. conjunction Indirect object Object Complement Clausals Clausals objects Secondary implicature Domain & Register Framenet Comment Sem. Class

Fig. 5. Pattern box in the CPA editor.

The pattern editor contains many features, here we mention only a few: 1) adding a new pattern, 2) renumbering the list of patterns (by drag & drop), 3) inserting a new pattern at a specific place, 4) merging two patterns, 5) accessing a filtered concordance of all the lines tagged with a specific pattern, etc. The editor is synchronized with Sketch Engine annotations periodically, such that every time a pattern is added or removed, the information is updated in both systems. The pattern editor comes with an ontology editor which enables the lexicographer to create, delete or update semantic type nodes in the CPA ontology. Several methods have been developed in order to access all the verbs which use a particular semantic type, and all the nouns matching a semantic type in the corpus (Figure 6).



- Anything *i.e. anything at all* edit details
  - Entity edit details
    - Abstract_Entity edit details
      - Concept *Must be a word meaning 'concept'; otherwise use Anything = Concept* edit details
        - Proposition edit details
          - Narrative edit details
        - Rule edit details
          - Permission edit details
        - Dispute edit details
        - Information edit details
      - Information_Source edit details
        - Document *[Information_Source, Artifact]* edit details
          - Agreement *[Speech_Act, Document]* edit details
        - Language edit details

Semantic Type: Attitude

Verb	Pattern number	Freq
arouse	1,2	763
resent	1	457
applaud	3	158
awaken	3,4,5	78
brush	1,5	76
repeat	1	72
trigger	2	55
greet	2	50
admit	6	39

Nouns	S	O	Ad	∑
glory	0	0	15	15
intention	0	10	0	10
determination	0	9	0	9
glow	0	0	7	7
willingness	0	7	0	7
stance	0	6	0	6
acceptance	0	4	0	4
attitude	0	3	0	3
readiness	0	3	0	3

Fig. 6. Ontology editor with populated nouns and patterns.

The JavaScript code of the editor was also used for Pattern Dictionary of English Prepositions by Ken Litkowski [5].

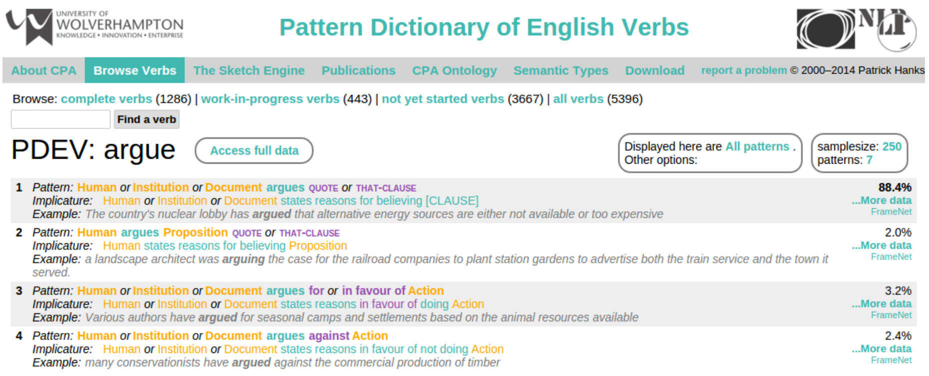
5 DEB platform

The CPA editor interacts with the database server via API (Ruby, WebBrick) which manages the CPA databases (one per language, currently English, Italian and Spanish). The server is based on the DEB[7] dictionary management server [6] and is based on private and public methods. Several servlet methods have been designed in order to ensure maximum efficiency as well as to limit server overload. The general mechanism is that a JavaScript client sends AJAX queries to the DEB server which queries the database and sends back responses in JSON. There are two main servlets, one for the CPA editor, one for the public version (Section 6). The server also maintains users and their privileges (e.g. only expert users can label a verb as complete in the editor, or modify them thereafter).

[7] `deb.fi.muni.cz`

6 Public access to PDEV

To provide access to the up-to-date data of PDEV and the ontology we have developed a user-friendly online tool[8] which is connected to the main PDEV database. It is structured similarly to the editor but the style has been entirely re-shaped and only a limited number of methods are available (it does not include pattern boxes for instance). The website uses colour codes for different pieces of information (semantic types, grammatical categories, lexical items). The public website interacts with Sketch Engine to access labelled examples for each pattern. A specific feature enables to show the best sentence example (GDEX, [7]) for each pattern in the pattern list (Figure 7). Ontology in the form of a semantic type list and a hierarchical structure is also available.



© 2000–2014 Patrick Hanks

Browse: complete verbs (1286) | work-in-progress verbs (443) | not yet started verbs (3667) | all verbs (5396)

PDEV: argue

Displayed here are All patterns .
Other options:

samplesize: 250
patterns: 7

1 *Pattern:* Human *or* Institution *or* Document argues QUOTE *or* THAT-CLAUSE — **88.4%**
Implicature: Human *or* Institution *or* Document states reasons for believing [CLAUSE] — **...More data**
Example: *The country's nuclear lobby has* ***argued*** *that alternative energy sources are either not available or too expensive* — FrameNet

2 *Pattern:* Human argues Proposition QUOTE *or* THAT-CLAUSE — 2.0%
Implicature: Human states reasons for believing Proposition — **...More data**
Example: *a landscape architect was* ***arguing*** *the case for the railroad companies to plant station gardens to advertise both the train service and the town it served.* — FrameNet

3 *Pattern:* Human *or* Institution *or* Document argues for *or* in favour of Action — 3.2%
Implicature: Human *or* Institution *or* Document states reasons in favour of doing Action — **...More data**
Example: *Various authors have* ***argued*** *for seasonal camps and settlements based on the animal resources available* — FrameNet

4 *Pattern:* Human *or* Institution *or* Document argues against Action — 2.4%
Implicature: Human *or* Institution *or* Document states reasons in favour of not doing Action — **...More data**
Example: *many conservationists have* ***argued*** *against the commercial production of timber* — FrameNet

Fig. 7. Public interface to PDEV data.

7 SemEval 2015 Task 15 dataset

In order to support NLP research in semantic parsing which would help to evaluate the impact of the CPA resources in semantic tasks, a high quality dataset derived from PDEV was produced and used in Task 15 at Semeval competition in 2015[9]. The goal of this task was to evaluate to which extent NLP systems could contribute to the creation of a lexicographical entry. To maximize participants' interest as well as to simplify this complex task, it was broken down into 3 inter-connected subtasks: 1) CPA parsing: all sentences in the dataset to be syntactically and semantically parsed. 2) CPA clustering: all sentences in the dataset to be grouped according to their similarities. 3) CPA pattern editing: all verb patterns found in the dataset to be described in terms of their syntactic and semantic properties.

[8] `www.pdev.org.uk`

[9] `alt.qcri.org/semeval2015`

Two datasets were created: Microcheck, which included all three tasks, and was intended to be used in analysing the correlation of the tasks; Wingspread, a larger dataset yet only including task 2 and 3, as task 1 required manual annotation of semantic and syntactic properties of arguments in the context of the verb. It is worth noting that this was the first attempt at annotating CPA pattern arguments in context. For more details, refer to SemEval paper [8]. The resource is available from the SemEval website[10].

While the task did not gather as much interest as desired, it fostered the development of a baseline system (mostly unbeaten by competing systems), which connected all three tasks together. This system was integrated into the CPA interface under the username "auto-cpa" for scrutiny and validation by CPA lexicographers.

8 LEMON API

It is important to release the data produced by CPA lexicographers for use by NLP developers, in the same fashion as WordNet, FrameNet and other resources.

The problem for CPA is that the data was scattered in different files, in different formats, spanning several database tables, each holding heterogeneous types of content (corpus examples, links, ontology, . . .).

The solution was to encode the resource using RDF as linked open data in a DEB server method having access to Sketch Engine annotation data. The name of this project is "PDEV-LEMON" and the first version was released in 2014 [9]. The server script connects elements of patterns stored in the database to the CPA ontology, and calls Sketch Engine API methods to retrieve annotated examples. The dictionary is encoded using the LEMON model [10] which provides the general structure and features to enable an easy instantiation of a lexicon using an ontology framework such as OWL. PDEV-LEMON includes 7 ontologies which describe the CPA semantic ontology as well as ontologies describing specific concepts and relations used in PDEV-LEMON, taxonomies of domains, and registers and so on. The first release of PDEV-LEMON included 17,634 triples, 3,702 patterns and 10,799 arguments.

A second release is planned for the end of 2015 and it will include a full linked data encoding of examples (which, in the first release linked back to the public access website) using NIF.[11] The conversion script was made in such a way as to handle other languages (Spanish and Italian so far), so it is expected that future releases will include dictionaries in languages other than English.

9 Conclusion

This report describes the software tools to support the development of pattern dictionaries applying the CPA method. Latest developments of the CPA

[10] `alt.qcri.org/semeval2015/task15`

[11] NLP Interchange Format: `persistence.uni-leipzig.org/nlp2rdf`

infrastructure have made it easier for lexicographers to quickly draft high quality dictionary entries, while being robust for a large number of simultaneous users. Several projects, such as PDEV-LEMON, and the development of the Microcheck and Wingspread datasets, have been launched to disseminate the resource and facilitate its use in NLP.

In parallel, this infrastructure was (and is being) also used to develop verb pattern dictionaries for Spanish [12] and Italian. In the future, if the number of languages increases (particularly with Czech, German and French), it should facilitate the painstaking lexicographical work needed to developed pattern dictionaries.

Terminology and language learning resources can also benefit from this suite of tools. Among future improvements of the CPA editor, we may mention providing support to connect patterns monolingually and across languages (English, Spanish, Italian). These links could substantially contribute to areas such as knowledge-based Machine Translation.

The CPA editor and the annotation in Sketch Engine are used on daily basis by lexicographers.

Acknowledgement We would like to thank lexicographers, annotators, trainees, testers, who have used these tools and provided feedback and to Patrick Hanks, the main user of these tools, for his unrelenting input and patience.

This work has been partly supported by an AHRC grant [DVC, AH/J005940/1, 2012–2015], and by the Czech Science Foundation under the project GA15-13277S, and by the project MUNI/A/1165/2014 Čeština v jednotě synchronie a diachronie – 2015.

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**(1) (2014) 7–36
2. Leech, G.: 100 million words of english: the british national corpus (bnc). Language Research **28**(1) (1992) 1–13
3. Rychlý, P.: Manatee/bonito-a modular corpus manager. In: 1st Workshop on Recent Advances in Slavonic Natural Language Processing, Faculty of Informatics (2007) 65–70
4. El Maarouf, I., Bradbury, J., Baisa, V., Hanks, P.: Disambiguating verbs by collocation: Corpus lexicography meets natural language processing. In: Proceedings of LREC, Reykjavik, Iceland (2014) 1001–1006
5. Litkowski, K.: Pattern dictionary of english prepositions. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Baltimore, Maryland, Association for Computational Linguistics (June 2014) 1274–1283

[12] `verbario.com`

6. Horák, A., Vossen, P., Rambousek, A.: A distributed database system for developing ontological and lexical resources in harmony. In: Computational Linguistics and Intelligent Text Processing. Springer (2008) 1–15
7. Kilgarriff, A., Husák, M., McAdam, K., Rundell, M., Rychlỳ, P.: Gdex: Automatically finding good dictionary examples in a corpus. In: Proceedings of the XIII EURALEX International Congress (Barcelona, 15-19 July 2008). (2008) 425–432
8. Baisa, V., Bradbury, J., Cinková, S., El Maarouf, I., Hanks, P., Kilgarriff, A., Popescu, O.: Semeval-2015 task 15: A cpa dictionary-entry-building task. In: Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), Denver, Co, USA (2015)
9. El Maarouf, I., Bradbury, J., Hanks, P.: Pdev-lemon: a linked data implementation of the pattern dictionary of english verbs based on the lemon model. In: Proceedings of the 3rd Workshop on Linked Data in Linguistics (LDL): Multilingual Knowledge Resources and Natural Language Processing at the Ninth International Conference on Language Resources and Evaluation (LREC'14), Reykjavik, Iceland (2014)
10. McCrae, J., Aguado-de Cea, G., Buitelaar, P., Cimiano, P., Declerck, T., Gómez-Pérez, A., Gracia, J., Hollink, L., Montiel-Ponsoda, E., Spohr, D., et al.: Interchanging lexical resources on the semantic web. Language Resources and Evaluation **46**(4) (2012) 701–719
11. Hanks, P.: How people use words to make meanings: Semantic types meet valencies. In Boulton, A., Thomas, J., eds.: Input, Process and Product: Developments in Teaching and Language Corpora. Masaryk University Press, Brno (2012) 54–69
12. Kilgarriff, A., Palmer, M.: Introduction to the special issue on senseval. Computers and the Humanities **34:1–2** (2000)
13. Hanks, P., Pustejovsky, J.: A pattern dictionary for natural language processing. Revue Française de linguistique appliquée **10:2** (2005)
14. Hanks, P.: Lexical Analysis: Norms and Exploitations. MIT Press, Cambridge, MA (2013)
15. Jezek, E., Magnini, B., Feltracco, A., Bianchini, A., Popescu, O.: T-pas; a resource of typed predicate argument structures for linguistic analysis and semantic processing. In: Proceedings of LREC, Iceland (2014)
16. Renau, I., Battaner, P.: Using cpa to represent spanish pronominal verbs in a learner's dictionary. In: Proceedings of the XV EURALEX, Norway (2012)
17. Hanks, P.: Corpus pattern analysis. In: Proceedings of the XI EURALEX, Lorient, France (2004)
18. Alonso Campo, A., Renau, I.: Corpus pattern analysis in determining specialised uses of verbal lexical units. Terminalia **7** (2013) 26–33
19. El Maarouf, I., Baisa, V.: Automatic classification of semantic patterns from the pattern dictionary of english verbs. In: Proceedings of JSSP2013, Trento, Italy (2013) 95–99
20. Bradbury, J., El Maarouf, I.: An empirical classification of verbs based on semantic types: the case of the 'poison' verbs. In: Proceedings of JSSP2013, Trento,Italy (2013) 70–74
21. Cinková, S., Holub, M., Kríž, V.: Optimizing semantic granularity for nlp - report on a lexicographic experiment. In: Proceedings of the 15th EURALEX International Congress, Oslo, Norway (2012) 523–531
22. Mills, C., Levow, G.A.: Cmills: Adapting srl features to dependency parsing. In: Proceedings of SemEval 2015, Denver, USA (2015)
23. Pedersen, T.: Duluth: Word sense discrimination in the service of lexicography. In: Proceedings of SemEval 2015, Denver, USA (2015)

24. Feng, Y., Deng, Q., Yu, D.: Blcunlp: Corpus pattern analysis for verbs based on dependency chain. In: Proceedings of SemEval 2015, Denver, USA (2015)
25. Cinková, S., Holub, M., Kríž, V.: Managing uncertainty in semantic tagging. In: Proceedings of 13th Conference of the European Chapter of the Association for Computational Linguistics, Avignon, France (2012) 840–850
26. Cinková, S., Holub, M., Rambousek, A., Smejkalová, L.: A database of semantic clusters of verb usages. In: Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012), Istanbul, Turkey (2012) 3176–3183
27. Holub, M., Kríž, V., Cinková, S., Bick, E.: Tailored feature extraction for lexical disambiguation of english verbs based on corpus pattern analysis. In: Proceedings of the 24th International Conference on Computational Linguistics (Coling 2012), Mumbai, India (2012) 1195–1209
28. Kawahara, D., Peterson, D.W., Popescu, O., Palmer, M.: Inducing example-based semantic frames from a massive amount of verb uses. In: Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics. (2014) 58–67
29. Popescu, O.: Building a resource of patterns using semantic types. In: Proceedings of LREC, Boston, USA (2012)
30. Popescu, O.: Learning corpus patterns using finite state automata. In: Proceedings of the 10th International Conference on Computational Semantics, Potsdam, Germany (2013) 191–203
31. Popescu, O., Palmer, M., Hanks, P.: Mapping cpa onto ontonotes senses. In: Proceedings of LREC, Reykjavik, Iceland (2014) 882–889
32. Pustejovsky, J., Hanks, P., Rumshisky, A.: Automated induction of sense in context. In: Proceedings of COLING, Geneva, Switzerland (2004)
33. Rumshisky, A., Hanks, P., Havasi, C., Pustejovsky, J.: Constructing a corpus-based ontology using model bias. In: Proceedings of FLAIRS, Melbourne, FL (2006) 327–332
34. Sinclair, J.: Corpus, concordance, collocation. Oxford University Press (1991)
35. Baker, C.F., Fillmore, C.J., Lowe, J.B.: The berkeley framenet project. In: Proceedings of the 17th international conference on Computational linguistics-Volume 1, Association for Computational Linguistics (1998) 86–90
36. Pustejovsky, J.: The Generative Lexicon. MIT Press (1995)
37. Fillmore, C.J.: Frames and the semantics of understanding. Quaderni di Semantica **6**(2) (1985) 222–254
38. Wilks, Y.: A preferential, pattern-seeking, semantics for natural language inference. Artif. Intell. **6**(1) (1975) 53–74
39. Kilgarriff, A., Rychlý, P.: Semi-automatic dictionary drafting. In de Schryver, G.M., ed.: Oxford Handbook of Innovation. Menha Publichers, Kampala (2010)
40. Litkowski, K.: Corpus pattern analysis of prepositions. Technical report, CL Research (02 2012)
41. El Maarouf, I., Alferov, E., Cooper, D., Fang, Z., Mousselly Sergieh, H., Wang, H.: The guanxi network: a new multilingual llod for language learning applications. In: Proceedings of NLP&LOD workshop, RANLP. (2015)
42. El Maarouf, I., Marsic, G., Orasan, C.: Barbecued opakapaka: Using semantic preferences for ontology population. In: Proceedings of RANLP. (2015)
43. Jezek, E., Hanks, P.: What lexical sets tell us about conceptual categories. Lexis **4**(7) (2010) 22
44. Pustejovsky, J., Jezek, E.: Semantic coercion in language: Beyond distributional analysis. Italian Journal of Linguistics **20**(1) (2008) 175–208

45. Popescu, O., Vo, N.P.A., Feltracco, A., Jezek, E., Magnini, B.: Toward disambiguating typed predicate-argument structures for italian. Proceedings of CLIC-IT14 (2014)
46. Feltracco, A., Jezek, E., Magnini, B.: Opposition relations among verb frames. In: Proceedings of the 3rd Workshop on EVENTS at the NAACL-HLT. (2015) 16–24

Semantic Regularity of Derivational Relations

Pavel Šmerk

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

Abstract. The paper presents a very preliminary attempts to employ the *word2vec* tool to evaluate semantic regularity within particular derivational relations in the data of the *Derivancze* derivational analyzer of Czech.

Keywords: derivational morphology, semantics of the derivational relations, word2vec

1 Introduction

In [1] we (together with Karel Pala) presented *Derivancze*, derivational analyzer for Czech. We gave reasons, why we prefer not to include all derivational relations, but to limit our data only to those relations which are semantically transparent and regular. But we had no objective measure which could assure us that our relations actually are semantically regular.

In this paper we present an attempt to employ *word2vec* [2,3,4] tool to evaluate the semantic regularity of derivational relations in our data. In the following section we shortly describe derivational relations of *Derivancze* and the data of the analyzer. In the next section we present our experiments and results, namely with some "averaged concept", which should represent the semantics of a given derivational relation. Finally we discuss some of the possible future improvements.

2 Derivational Relations and Data of *Derivancze*

The following description of the derivational relations is taken from our previous paper [1] (refer to it for more information):

– `k1verb`, `k2pas`, `k2proc`, `k2rakt`, `k2rpas`, and `k2ucel` from verbs, where:
  - `k1verb` derives nouns describing process, action or state denoted by the verb (*kropit–kropení*; *sprinkle–sprinkling*),
  - `k2pas` and `k2rpas` are passive participle and past passive adjectival participle, i. e. two forms of adjectives which describe the patient or object of the action (*kropit–kropen / kropený*; *sprinkle–sprinkled*),

  - `k2proc` derives present active adjectival participles, i. e. adjectives describing a subject doing the action (*kropit–kropící*; *sprinkle–sprinkling* (man)),
  - `k2rakt` are past active adjectival participles, i. e. adjectives describing subjects which have completed the action (*pokropit–pokropivší*; *sprinkle–who has springled st.*), and
  - `k2ucel` derives adjectives which describe an object used for the action (*kropit–kropicí*; *sprinkle–sprinkling* (machine)),
- verb → agent noun relation `k1ag` (*bádat–badatel*; *research–researcher*),
- adjective → name of the property relation `k1prop` (*rychlý–rychlost*; *fast–speed*),
- adjective → adverb relation `k6a` (*dobrý–dobře*; *good–well*),
- noun → possessive adjective relation `k2pos` (*otec–otcův*; *father–father's*),
- noun → relational adjective relation `k2rel` (*virus–virový*; *virus–viral / virus*), semantically perhaps the most heterogenous relation among the Derivancze relations,
- relations `k1f`, `k1jmf`, and `k1jmr` express changes in gramatical gender:
  - `k1f` derives feminines from general masculines (*doktor–doktorka*; *doctor*$_{\text{MASC}}$ *–doctor*$_{\text{FEM}}$),
  - `k1jmf` derives feminine forms of surnames (*Novák–Nováková*), and
  - `k1jmr` derives family forms of surnames (*Novák–Novákovi*) — it should be noted that `k1f` and `k1jmf` cannot be joined because of names of nationalities, which can also act as surnames, but the derived forms differ (*Rus–Ruska* X *Rusová*, i. e. *Russian*$_{\text{FEM}}$ X *Mrs. Rus*),
- area or city → inhabitant name relation `k1obyv` (*Kanada–Kanaďan*; *Canada–Canadian*), formally the most heterogenous relation in Derivancze,
- noun → deminutive relation `k1dem` (*dům–domek*; *house–little house*).

The Table 1 shows a distribution of the derivational pairs in *Derivancze* data according to the derivational relation. The row `var` is the semantic equivalence (see [1] for details). The numbers are without the pairs in which both members are the same[1]. The column **total** is the total number of pairs in our data. The columns **N+** are numbers of pairs whose both members occur in the corpus CzTenTen [5] (ca 5.3 billion tokens) with a frequency at least N.

3 Experiments

One of the possible benchmarks for evaluating the word vectors produced by the *word2vec* tool is complementing triplets of form for instance

Greece Athens Norway ?
Kazakhstan Astana Zimbabwe ?
. . .

[1] For instance, the passive participle (`k2pas`) of the verb *přejet* is *přejet*.

Table 1. Distribution of derivational pairs according to relation.

Relation	total	1+	10+	50+	1000+
k1ag	703	400	400	248	89
k1dem	6334	4122	4122	2983	1218
k1f	3196	1800	1800	1253	413
k1jmf	2211	1893	1893	1735	442
k1jmr	2207	1112	1112	427	31
k1obyv	261	215	215	168	91
k1prop	9902	5708	5708	3775	1110
k1verb	35723	15848	15848	11186	3906
k2pas	34779	7334	7334	4567	1249
k2pos	30936	7110	7110	3853	539
k2proc	15765	5017	5017	3462	1054
k2rakt	18106	369	369	101	9
k2rel	23518	15343	15343	10972	4045
k2rpas	35015	12377	12377	8177	2772
k2ucel	1672	1402	1402	1081	370
k6a	45108	11399	11399	6615	2108
var	1136	592	592	390	112
total	266572	92041	92041	60993	19558

(according to [2]). These triplets represent questions like "what is the word that is similar to Norway in the same sense as Athens is similar to Greece?" and the model should predict *Oslo* and *Harare*, respectively. The method, evaluation data and results are thoroughly described in many publications, e. g. [2,3,4].

In fact, we are not interested in some abstract similarity of *Greece* and *Athens* but these words are only particular representants of some common question "what is the capital city of ...". Interestingly, the results are much better if instead of a particular representant we use an average of more such examples.

3.1 "Averaged Concept"

As an illustration we present results of a very trivial experiment. We took the first 50 most frequent pair from `k1f` relation except the pair *muž/žena* (*man/woman*) and for these pairs and the word *muž* we evaluated questions in the aforementioned sense, so the answer should always be *žena*. We do the tests on models computed from 120 million, 1 billion, and 5.3 billion tokens from the corpus CzTenTen (the last is the whole corpus). Then we computed the average of these first 50 pairs and evaluated the question with these averages. The results are in the Table 2.

The numbers are always the cosine distance and can be interpreted as a measure of similarity (for further details again refer to [2,3,4]). For the first

Table 2. Experiment with an "averaged concept" representing `k1f`.

	120M		1G		5.3G	
avg	0.617013		0.744693		0.775647	
min	0.434191	občan/občanka	0.509271	občan/občanka	0.541018	občan/občanka
max	0.729597	otec/matka	0.857678	táta/máma	0.889231	táta/máma
1	**0.799218**	žena	**0.877726**	žena	**0.890839**	žena
2	0.648836	dívka	0.754475	dívka	0.762988	dívka
3	0.503029	ženský	0.610383	ženský	0.623256	chlapec
4	0.471445	otrokyně	0.595448	chlapec	0.621017	ženský
5	0.467925	matka	0.573623	matka	0.600329	mužův
6	0.467344	chlapec	0.564603	děvče	0.595389	dívenka
7	0.467089	milenec	0.555519	mladík	0.591686	děvče
8	0.466913	mladík	0.550952	mužův	0.582920	matka
9	0.462479	tmavovlasý	0.546331	partnerka	0.575861	partnerka
10	0.461867	chlap	0.541432	dívenka	0.572923	družka

50 pairs we present an average, minimum[2] and maximum[3] distance. The following ten rows are the top ten best answers for the "averaged" question. It can be easily seen that the pattern is the same for all three sizes of training data: the "averaged" right answer is always better than the result of any single pair, always much better than the average of the single pairs and always much better than any other possible answer — it should be added that *dívka* (as well as *dívenka* and *děvče*) is *young woman*, which means that even the second best answer is rather acceptable. The interesting is that all these effects are stronger for smaller data.

3.2 Evaluation of Derivational Relations

It suggests that for evaluation of semantic regularity of derivational relations we should try to find the concepts represented by the particular relations and check all pairs against that concept. As a very preliminary experiment we took the first 50 most frequent pairs of each derivational relation[4], compute the vector of this averaged concept and compare it with the first 150 most frequent pairs. It was evaluated on the whole corpus CzTenTen. The results are in the left half of the Table 3: the TOP10 column is the number of pairs where the correct answer was among the first 10 words according to the model (the whole vocabulary of the model has almost 1.5 million entries), the TOP1 column is the number of pairs where the correct answer was the first, the rank column is the average rank of the correct answer, and the distance is the average distance.

[2] In Czech *občanka* is not only a feminine form of *citizen* but also *identity card*.

[3] *otec* is *father*, *matka* is *mother*, *táta* is *daddy*, and *máma* is *mummy*.

[4] We do not explore `k1jmf` and `k1jmr` because they represent relations between proper names (surnames) and the automatic lemmatisation of the corpus is unfortunately of a very low quality for these words.

Table 3. Towards the concept of particular derivational relations.

	TOP10	TOP1	rank	distance	TOP10	TOP1	rank	distance
k1ag	53	6	4.42	0.595907	47	3	3.81	0.613398
k1dem	88	39	1.57	0.709737	85	40	1.59	0.718124
k1f	123	95	0.59	0.783001	123	98	0.56	0.786888
k1obyv	133	85	1.06	0.699675	133	86	1.00	0.703276
k1prop	88	30	1.69	0.654496	89	35	1.74	0.659931
k1verb	130	59	1.63	0.708456	127	56	1.83	0.712785
k2pas	134	99	0.59	0.828821	131	78	0.81	0.792059
k2pos	102	61	1.10	0.744492	95	51	1.36	0.740843
k2proc	119	79	1.14	0.738209	118	76	1.14	0.738124
k2rakt	44	3	3.75	0.563547	47	3	3.83	0.566550
k2rel	122	62	1.12	0.717147	123	56	1.57	0.713637
k2rpas	127	38	1.98	0.719895	124	40	2.06	0.719619
k2ucel	50	1	4.78	0.632999	52	1	4.71	0.636890
k6a	118	53	1.59	0.627307	116	65	1.31	0.635010

The second step was an attempt to refine the concept by selecting the best 50 pairs according to the initial concept. The results are in the right half of the Table 3. As there were many oddities in the data we calculated the average ranks and distances (in both halves of the Table 3!) only from the "TOP10" pairs, because it is OK for a wrong pair to be less similar to the refined concept, but the limit 10 is entirely arbitrary for the present.

4 Conclusion and Future Work

The results in the previous section show that word vectors computed by *word2vec* are useful for checking the semantic consistency of the derivational relations, at least to some extent. The number of pairs in TOP10 or even TOP1 is consistently rather high and many of the pairs which were not "successful" were either clearly wrong, e. g. *plat* (*salary*), *plátek* (*slice*) for `k1dem`, or with some very irregular semantics, e.g. *věřit* (*believe*), *věřitel* (*creditor*) for `k1ag` where the second is derived from the first, but it is very uncommon to say that *věřitel věří* (*creditor believes*).

In the future we plan to improve lemmatisation of the corpus, as the experiments revealed many systematic errors which obviously degrade the results. Both *word2vec* and the setup of the experiments have many parameters which were set almost arbitrarily and we will try to fine tune them. The refinement of the average concept was successful for some of the relations, but not for all, and although it was successful in the average, the improvement was much smaller than expected, thus we believe there is also a space for a further improvement. Then, of course, the refinement iteration should continue up to some fix point. The hugest amount of manual work will be demanded by removal of wrong pairs from the data. After all, we will be able to offer not only the derivational pairs alone, but also some additional information how close is the semantics of the particular pair to the average semantics of the respective derivational relation.

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

References

1. Pala, K., Šmerk, P.: Derivancze — Derivational Analyzer of Czech. In Král, P., Matoušek, V., eds.: Text, Speech, and Dialogue. Volume 9302 of Lecture Notes in Computer Science., Springer (2015) 515–523
2. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR **abs/1301.3781** (2013)
3. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In Burges, C.J.C., Bottou, L., Ghahramani, Z., Weinberger, K.Q., eds.: Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States. (2013) 3111–3119
4. Mikolov, T., Yih, W., Zweig, G.: Linguistic regularities in continuous space word representations. In Vanderwende, L., III, H.D., Kirchhoff, K., eds.: Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA, The Association for Computational Linguistics (2013) 746–751
5. Jakubíček, M., Kilgarriff, A., Kovář, V., Rychlý, P., Suchomel, V.: The TenTen Corpus Family. International Conference on Corpus Linguistics, Lancaster (2013)

Part IV

Logic, Semantic and Syntactic Processing

AST: New Tool for Logical Analysis of Sentences based on Transparent Intensional Logic

Marek Medveď and Aleš Horák

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

Abstract. Logical analysis of natural language is able to extract semantic relations that follow the underlying logical formalism. Transparent Intensional Logic (TIL) has been designed to capture even high-order relations between sentence elements and systematically work with all kinds of language references, i.e. extensions, intensions and hyperintensions.
In this paper, we introduce the first version of a new tool, called AST, for automatic semantic analysis of sentence. This tool is based on the TIL logic processing rules as they were implemented in the SYNT parser, in its logical analysis module. AST thus shares lexicons and semantic rules in the same format as in the SYNT parser, but allows to build upon the output of other syntactic parsers. AST is designed as a universal semantic analysis tool, which strictly separates the application logic and input data and strives for language independent analysis.
Within the evaluation, we present preliminary results of testing AST on selected problematic phenomena, which were not correctly processed by the SYNT logical analysis.

Keywords: semantics; semantic analysis; logical analysis; Transparent Intensional Logic; TIL

1 Introduction

Full semantic analysis of natural language (NL) texts still remains an open problem, although the problem is being partially solved from different points of view. The most comprehensive semantic systems build upon a mathematically sound formalism of a selected logical system. Mostly due to computability and efficiency, current systems work with the first order logic (or its variant). However, the low-order logic is not appropriate for capturing higher-order phenomena that occurs in natural language, such as belief attitudes, direct speech, or verb tenses [6].

In the following text, we present a new tool for automatic semantic analysis (AST) that emerged from (a module of) the Czech syntactic parser SYNT [3]. AST is now available as a standalone tool that is independent from the SYNT parser. AST works with the same input files (lexicons, semantic rules, ...) that

```
<tree>
{##start##
  {start
    {ss
      {clause
        {VL<leaf><idx>0</idx><w>Jedl</w>
         <l>jíst</l><c>k5eAaIgMnS</c></leaf>}
        {intr
          {adjp
            {ADJ<leaf><idx>1</idx><w>pečené</w>
             <l>pečený</l><c>k2eAgNnSc4</c></leaf>}
          }
          {np
            {N<leaf><idx>2</idx><w>kuře</w>
             <l>kuře</l><c>k1gNnSc4</c></leaf>}
          }
        }
      }
    }
    {ends
      {'.'<leaf><idx>3</idx><w>.</w><l>.</l><c>kX</c></leaf> }
    }
  }
}
</tree>
```

Fig. 1. Syntactic tree – text markup for "Jedl pečené kuře." (He ate a roasted chicken.)

were designed and developed in SYNT. AST can thus provide the semantic analysis in the form of Transparent Intensional Logic (TIL) constructions [1] independently on the input syntactic parser and language. Processing new language thus consists in a specification of four lexicon files that describe lexical items, verb valencies, prepositional valencies and a semantic grammar.

In the following sections, we describe the structure of the system, the language dependent files and the form of required input as processed by a syntactic parser.

2 The AST System

In this section, we introduce the main parts of the AST system, describe the content of language dependent files and formalize the required input of AST.

2.1 The AST Input

To create a semantic structure of a sentence, AST needs the output from previous NL analysis levels. A usual output is in the form of a syntactic tree

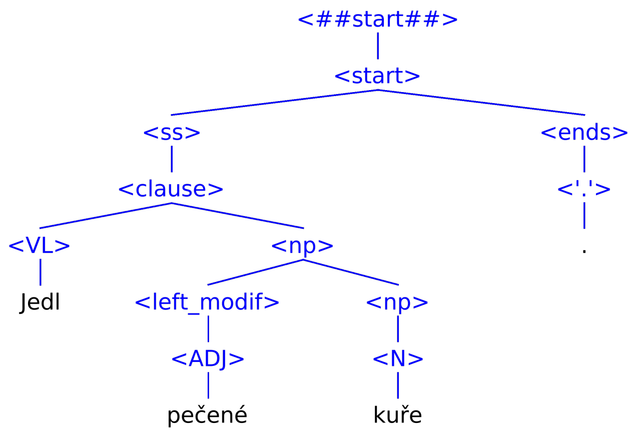


Fig. 2. Syntactic tree – visual for "Jedl pečené kuře." (He ate a roasted chicken.)

as provided by a syntactic parser. See Figure 1 for an example of a syntactic tree that is accepted as the AST input. The corresponding graphical tree representation is in Figure 2.

Besides the tree nodes and edges, the tree contains morphological information about each word: a lemma and a PoS tag [4], which are used by AST for deriving implicit out-of-vocabulary type information.

2.2 Language Dependent Files

The AST system itself is universal and can be used for semantic analysis of any language. However the main system core also uses input files that are language dependent and that need to be modified for addition of another language. In this section, we describe the format of those files needed to build the resulting logical construction.

The Semantic Grammar The resulting semantic construction is built by bottom-up analysis based on the input syntactic tree provided by the syntactic parser and by a semantic extension of the actual grammar used in the parsing process. To know which rule was used by the parser, AST needs the semantic grammar file. This file contains specification of semantic actions that need to be done before propagation of particular node constructions to the higher level in the syntactic tree. The semantic actions define what logical functions correspond to each particular syntactic rule. For instance, the `<np>` node in Figure 1 corresponds to the rule and action:

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

```
rule_schema: 2 nterms, '[#1,#2]'
1, 3, +np -> . left_modif  np . @level 0
  nterm 1: 1, 2, +left_modif -> . left_modifnl  . @level 0, k2eAgNnSc4
       TIL: $^0$pečený...$((o\iota)_{\tau\omega}(o\iota)_{\tau\omega})$
  nterm 2: 2, 3, +np -> . N  . @level 0, k1gNnSc4
       TIL: $^0$kuře...$(o\iota)_{\tau\omega}$

Processing schema with params:
    #1: $^0$pečený...$((o\iota)_{\tau\omega}(o\iota)_{\tau\omega})$
    #2: $^0$kuře...$(o\iota)_{\tau\omega}$
Resulting constructions:

    [$^0$pečený/$((o\iota)_{\tau\omega}(o\iota)_{\tau\omega})$, $^0$kuře/$(o\iota)_{\tau\omega}$]...$(o\iota)_{\tau\omega}$
```

Fig. 3. Analysis of the expression "pečené kuře" (roasted chicken).

which says that 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. An example of processing such grammar rule is in Figure 3.

TIL Types of Lexical Items The second language dependent file defines lexical items and their TIL types. The types are hierarchically built from four simple TIL types [2]:

- o: representing the truth-values,
- ι: class of individuals,
- τ: class of time moments, and
- ω: class of possible worlds.

AST contains rules for deriving implicit types based on PoS tags of the input words, so as the lexicons must prescribe the type only for cases that differ from the implicit definition. A lexical item example for the verb "jíst" (eat) is:

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

The exact format of the lexical item in the input file is as follows: the lemma starts on a separate line. After the lemma there is a list of lines where an (optional) POS tag filter precedes the resulting object schema (here `otriv`, i.e. *o-trivialisation*) and TIL type (here *verbal object with one ι-argument*).

Verb Valencies The next language dependent file is a file that defines verb valencies and schema and type information for building the resulting construction from the corresponding valency frame. An example for the verb "jíst" (eat) is as follows

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

This record defines the valency of <*somebody*> *eats* <*something*>, given by the *brief valency frame* `hPTc4` of the *object* (an animate or inanimate noun phrase in accusative), and the resulting construction of the *verbal object* (`V(v)`) derived as an application of the *verb* (`#0`) to its argument (the sentence object) with possible extensification (`try(#1)`) and the appropriate possible world variable (`V(w)`).

Prepositional Valency Expressions The last file that has to be specified for each language is a list of semantic mappings of prepositional phrases to valency expressions based on the head preposition. The file contains for each combination of a preposition and a grammatical case of the included noun phrase all possible valency slots corresponding to the prepositional phrase. For instance, the record for the preposition "k" (to) is displayed as

```
k
3 hA hH
```

saying that "k" can introduce prepositional phrase of a *where-to* direction `hA` (e.g. "*k lesu*" – "to a forest"), or a modal *how/what* specification `hH` (e.g. "*k večeři*" – "to a dinner").

2.3 System parts

The AST system is implemented in the Python 2.7 programming language and consists of six main parts:

– the *input parser*: reads standard input, extracts tree structures and creates tree object for each tree from input,
– the *grammar parser*: reads the grammar file and assigns a grammar rule and appropriate actions to each node inside the tree,
– the *lexical item parser*: reads the file with lexical item schemata and TIL types and assigns the type to each leaf in the tree structure,
– the *schema parser*: according to a logical construction schema coming with a semantic action, this module creates a construction from sub-constructions,
– the *verb valency parser*: picks up the correct valency for given sentence and triggers the schema parser on sub-constructions according to the schema coming with the valency, and
– the *prepositional valency expression parser*: reads the possible valency expressions assigned to prepositional phrases used as (optional) valency slots in the actual sentence valency frame.

3 Error Analysis

During the AST development, AST is continuously evaluated in comparison with the original SYNT TIL logical analysis. In these tests, a number of uncovered phenomena is described and implemented in AST. The number of NL constructs covered by the AST logical analysis is thus still growing and, already in the current version, surpasses the original logical analysis.

In the following paragraphs, we present the results of error analysis of selected 200 sentences and the corresponding problems related to the original analyses of these sentences.

3.1 Sentences with Two Items Divided by "and"

The construction for the following sentence:

Vidíte zásadnější rozdíly mezi přístupy českých a západních informačních firem?
(Can you see the main difference between Czech and west information companies?)

is (schematically) analysed as `[západní, [český, x8]] and [firma, x8]`.[1] This means that both words "*západní*" (west) and "*český*" (Czech) are modifiers of the word "firma" (company) which is not correct. The correct analysis for this sentence is `[západní, x8] and [firma,x8] and [český, x9] and [firma, x9]`.[2]

3.2 Verb Valency vs. Clause Valency

The concluding clause semantic construction is created according to the content of the syntactic tree of the clause. The clause valency (valencies) is built from the subtrees, and in the next step the clause valency is matched to the verb valency from lexicon and the schema assigned to the verb valency is used for creating the clause logical construction. However, if the system creates an incorrect clause valency or the verb valency has no suitable option, the clause construction is not created and the resulting semantic analysis is void.

In the analysis of the following sentence

Možná, že se tito lidé ani nesetkali.
(Maybe, these people never met each other.)

the clause schema does not contain the reflexive pronoun "*se*"[3], so the system finds only the verb valency for the word "*setkat*" (meet), which does not contain suitable schema for semantic analysis.

[1] transl. `[west, [Czech, x8]] and [company, x8]`.
[2] transl. `[west, x8] and [company,x8] and [Czech, x9] and [company, x9]`.
[3] here in the meaning of "each other"

Table 1. 200 sentences evaluated by the SYNT TIL system and the AST system.

system	correct	correct in %	incorrect	incorrect in %
SYNT TIL	131	65.5 %	69	35.5 %
AST	158	**79.0 %**	42	**21.0 %**

3.3 Verb Valency Schema Update

In some cases, the clause valency is created correctly but the verb valency file does not contain an option that can match with the created cause valency. To solve this type of problems, AST files must be updated to add new option for the missing verb valency to create the correct semantic analysis.

3.4 Verb Valency Schema Missing

The verb valency list is created from the Czech VerbaLex lexicon [5], which is a large database of more than 10,000 Czech verb lemmata and their verb frames. However, it can happen that some verbs are not included in VerbaLex thus the system does not contain the verb frame.

In the test data, there were three verbs that are not included in VerbaLex, that it why they have been added to the AST verb list.

3.5 System errors

The previous SYNT TIL analysis contains a construction checker that does not allow the dash character "-" in the name of an object construction, such as `[0(Si-an/`ι`)]` (a proper name). In such case, the construction is rejected and the semantic analysis fails. The AST analysis allows such object naming, so the analysis can continue successfully.

4 Evaluation

In this section, the AST system is compared with the original SYNT TIL logical module. For evaluation, we have picked up 200 randomly chosen sentences that were processed by both systems and the resulting analysis for each sentence was manually checked. For the final results, that shows 14 % improvement of AST to SYNT TIL, see Table 1.

5 Conclusions

In this paper, we have introduced new language and parser independent tool for semantic analysis, called AST, that is based on the SYNT TIL logical analysis. The new AST system is designed as a lightweight standalone module, that can be straightforwardly updated and improved. Already in the first version AST corrects several frequent errors of its predecessor, and presents a 14 % increase in the number of correctly analyzed sentences.

This new implementation of semantic analysis brings the necessary simplicity for future development and parser and language independence.

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

References

1. Duží, Marie and Jespersen, Bjorn and Materna, Pavel, Procedural Semantics for Hyperintensional Logic: Foundations and Applications of Transparent Intensional Logic, Springer Science & Business Media (2010)
2. Horák, Aleš: Types in Transparent Intensional Logic and Easel – a Comparison, Proceedings of the IASTED International Conference Artificial Intelligence and Applications, 833–837 (2004)
3. Horák, Aleš and Jakubíček, Miloš and Kovář, Vojtěch: Linguistic Logical Analysis of Direct Speech, RASLAN 2012 Recent Advances in Slavonic Natural Language Processing, 51–59 (2012)
4. Jakubíček, Miloš and Kovář, Vojtěch and Šmerk, Pavel: Czech Morphological Tagset Revisited, Proceedings of Recent Advances in Slavonic Natural Language Processing, 29–42 (2011)
5. Nevěřilová, Zuzana and Grác, Marek : Common Sense Inference using Verb Valency Frames, In Proceedings of 15th International Conference on Text, Speech and Dialogue, 328–335 (2012)
6. Tichý, Pavel: The Foundations of Frege's Logic. de Gruyter, Berlin, New York (1988)

Annotation of Multi-Word Expressions in Czech Texts

Zuzana Nevěřilová

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

Abstract. Multi-word expressions (MWEs) are difficult to define and also difficult to annotate. Some of them cause serious errors in the traditional annotation pipeline tokenization – morphological analysis – morphological disambiguation. Many cases of incorrect annotation in Czech corpora are known. To narrow the research topic, we focus only in fixed MWEs – those with fixed word order and no ellidable components. In this paper, we propose a corpus-based method that reveals fixed MWE candidates. From the web-based corpus of Czech, we extracted 25,091 expressions, 2,140 of them were identified as MWEs, 332 as probable MWEs, and 174 of them can be either MWEs or one single word.
Our method is based on corpus data observation that indicates that people are unsure when writing a MWE whether it is one word, a word with dashes, or several words. The result is a list of MWE candidates and also an application that classifies the input as MWE, probable MWE, or non-MWE.

Keywords: multi-word expressions, corpus, orthographical variants

1 Introduction

Most corpora are single-word tokenized, i.e. the input text is segmented on single tokens (surrounded by white spaces or punctuation). This common starting point sometimes causes problems in subsequent text analysis.

For example, the expression *a priori* is tokenized in two tokens: *a* and *priori*. The token *a* is ambiguous in Czech (it means *and* among others), and *priori* is not a Czech word. The expression is widely used in Czech (1.10 per million in Czech corpus `czTenTen`) but the tagging in `czTenTen` and `syn2010` (one part of the Czech National Corpus) is incorrect: *a* is tagged as conjunction, *priori* is tagged as adverb in both corpora.

To solve this problem, a large dictionary of multi-word expressions (MWEs) that are problematic for tagging would be useful. [7] define MWEs as "idiosyncratic interpretations that cross word boundaries (or spaces)" and distinguish three classes: fixed MWEs, semi-fixed MWEs (such as compound nominals or proper names), and syntactically flexible expressions (such as idioms).

Although many types of MWEs exist, in this work, we focus on two types of fixed MWEs:

– fixed MWEs (also called frozen MWEs or words with spaces): e.g. *a priori*, *křížem krážem* (meaning *(travel) the length and breadth*)
– *almost* fixed MWEs (e.g. those with inflectional component and no ellidable component): e.g. *hot dog* which has inflected forms such as *hot dogy*, *hot dogu* etc., *Karel IV.* (meaning the king Charles IV) with inflected forms such as *Karla IV.*, *Karlu IV.* etc.

Both types have fixed word order and no ellidable component. These criteria differ the studied MWEs from other types. For example, *New York Rangers* is a MWE but it has ellidable component since we can find *Rangers* in texts in the meaning of *New York Rangers*.

In order to reduce tagging errors, we need to distinguish problematic MWEs from all other n-grams. We did several measurements on MWEs found in Czech Wiktionary, n-grams with high associative measures, and a set of random n-grams. We propose a method to distinguish fixed MWEs from all other collocations. Such MWEs can be stored in a dictionary with the appropriate grammatical information (part-of-speech and grammatical categories).

We plan to use our method to re-annotate Czech corpus *cztenten* [8].

1.1 Current Tokenization and Tagging of Czech Corpora

Currently, Czech corpora developed at NLPC are annotated automatically using the `unitok` tokenizer and the morphological analyzer `majka` [10]. In case of words unknown to `majka`, the word tag guesser is used. Afterwards, all possible lemmata and morphological tags are disambiguated using the tagger `desamb` [9] based on inductive logical programming.

The most problematic part of this process is the tagging of unknown words and cross-lingual homonyms. In the former case, the guesser proposes possible lemmata and tags regardless of the context (the context is used only in the disambiguation part). In the latter case, Czech homonyms are used even when their frequency is very low. For this reason, the corpus czTenTen contains 17,261,404 imperatives (3,405.00 per million) which is in reality very unlikely. Rather than imperatives, a significant part of these tokens are foreign words. For example, the English word *top* is homonymous to the Czech imperative from *to drown*. Most expressions such as *Top 10* are therefore annotated as imperative followed by a number.

1.2 Paper Outline

In the Section 2, we describe the MWE fixedness issue in order to define MWE as most precisely as possible, Section 3 shows our preliminary research. In Section 4, we present the new MWE discovery method, Section 5 shows the results, and Section 6 discusses them. Section 7 proposes future work.

2 Related Work

Extraction of MWEs has been widely studied in many languages and several techniques exist. Statistical techniques are based on measuring co-occurence of tokens by various measures, e.g. t-score or pointwise mutual information [2], logDice [6] or mean and variance [3]. All techniques generally lead to a list of MWE candidates. However, finding fixed or semi-fixed candidates requires testing of basic properties of MWEs.

Usually, MWEs are identified by three criteria:

– non-compositionality: the meaning of the MWE is not perceivable by meanings of its components
– non-modifiability: the MWE can be used e.g. only in singular, inflectional languages distinguish between absolute non-modifiability and limited modifiability
– non-substitutability: the components of the MWE cannot be substituted by their synonyms

[4, p. 24] adds another criterion – the asymmetric association: "lexical association between components is much stronger from one component to another than vice versa". [4] shows that asymmetry is very important in case of noun phrases.

Testing the appropriateness of a MWE candidate includes testing the criteria, e.g. [1] tested non-compositionality, [5] tested non-substitutability of MWE candidates.

Multi-word tokenization has been deeply studied by [4].

3 A Preliminary Study

In our approach, we narrow the wide set of MWEs only to those that have problematic tagging, i.e. contain a non-Czech word or are a fixed sentence (a routine formula or a named entity). The hypothesis is that such MWEs are strongly fixed: either they are frozen (such as *křížem krážem* meaning *criss-cross*), or almost-frozen: they have fixed word order and only some components are subject of inflection (such as *hot dog*).

In the preliminary study, we did not focus much on MWE extraction. First, we only examined collocations from Wiktionary page concerning word expressions[1]. We observed the modifiability and asymmetry of the MWEs extracted from Czech Wiktionary. We also noticed that the orthography of MWE is sometimes difficult. Language users are probably unsure whether a frozen MWE is one word or several words. Therefore, many frozen MWEs occur in corpora as one word or as one word with dashes. Our observations have shown that this feature discriminates frozen MWE that are often subject of incorrect annotation.

[1] `http://cs.wiktionary.org/wiki/Kategorie:Česká_slovní_spojení`

Table 1. MWEs that were found in the corpus as one word or one word with dashes. Significant asymmetry or significant number of occurrences as one word or one word with dashes are marked with bold.

expression	freq	asymmetry	one word	dash
a to	1000000	1.0000	**3669**	**23**
Abú Dhabí	625	0.1025	1	**9**
Abú Zabí	1869	0.2335	3	**4**
ad acta	454	**0.0075**	**22**	3
ad hoc	5593	**0.0773**	**251**	**1266**
ano i	3731	1.8137	**54**	**12**
čáry máry	539	**0.0328**	**80**	**51**
český jazyk	8096	0.5621	3	**49**
chtě nechtě	10846	1.1990	**76**	**799**
dejme tomu	28626	**30.2847**	**370**	**13**
domino efekt	171	**67.3155**	**12**	**21**
ex officio	202	**0.0080**	2	**9**
ex offo	848	**0.0317**	**6**	**136**
fata morgána	387	0.3592	**123**	**9**
faux pas	3091	**13.1264**	**30**	**624**
hned tak	19743	1.0000	**223**	**7**
Hradec Králové	58366	1.8396	**18**	3
i když	1000000	1.0000	**74121**	**82**
IP adresa	6205	0.5918	**26**	**103**
IQ tykve	849	**0.0647**	**13**	**20**
Kanárské ostrovy	3945	11.8394	2	**4**
Karlovy Vary	43242	0.6917	**37**	**18**
křížem krážem	4633	0.2343	**44**	**55**
lážo plážo	735	1.0676	**69**	**138**
mírnix týrnix	36	1.1222	**12**	**10**
mírnyx týrnyx	41	1.5405	**11**	**5**
na shledanou	5379	**0.0086**	**12784**	3
New York	61917	0.2749	**175**	**43**
nomen omen	479	1.1692	**8**	**41**
obchodní dům	5819	1.2479	**8**	**169**
Pán Bůh	14501	3.4631	**11539**	**4**
po o	1166	1.0000	**908**	**49**
po spa	14	**0.0037**	**777**	**4**
s to	18286	1.0000	**223647**	**4**
San Francisco	5889	0.0989	**12**	**5**
San Marino	2305	**0.0379**	**7**	3
Srí Lanka	2782	0.3019	**20**	**23**
techtle mechtle	263	0.9644	**28**	**52**
to do	96790	1.0000	**1278**	**347**
volky nevolky	378	0.9077	2	**16**
zatím co	13675	1.0000	**667751**	**52**
zuby nehty	7283	0.3530	**76**	**217**

The observations can be seen in Table 3. It shows bigrams from Czech Wiktionary that were also found as one word in the corpus `czTenTen` (with frequency min. 3) and as one word with a dash. It can be seen that asymmetry is not a determining feature in this case and in further research, we did not take it into account.

It can also be seen that some of the bigrams (český jazyk, Karlovy Vary, Kanárské ostrovy, obchodní dům) are noun phrases formed by an adjective and a noun in grammatical agreement. Such cases are not problematic to annotate. Thus, for further research, we probably have to add this syntactic criterion as well.

4 Methods

From the preliminary observations, we concluded that fixed MWEs are characterized by orthographic variability. They can be written as two words, one word, or one word with dashes.

The second step was to find such occurrences in Czech corpora and to examine whether or not they are fixed MWEs. We tried to find more MWEs again in the corpus `czTenTen`. We started with words with dashes. We found 6,296,839 occurrences of words with dashes, 2,029,715 unique occurrences, and 388,364 appearing more than once, and 205,708 occurrences appearing more than twice.

We categorized the occurrences of words with dashes in several categories:

1. compound adjectives (such as *česko-německý*, meaning Czech-German). These words are recognized correctly by the morphological analyser `majka`
2. abbreviations (such as *KDU-ČSL*, *DVB-T*, *CD-ROM*)
3. proper names (such as *Aix-en-Provence*, *Mercedes-Benz*, *Saint-Exupéry*, *Müller-Thurgau*)
4. chemical nomenclature (such as *beta-karoten*, *L-karnitin*, *B-komplex*)
5. originally English words occurring ordinarily in Czech, sometimes even with inflection (*e-mail*, *play-off*, *sci-fi*, *know-how*, *pop-music*, *set-top-box*, *line-up*). Some of them are recognized by the morphological analyser `majka`, some of them are not in its database.
6. other words of foreign origin (*kung-fu*, *au-pair*, *tee-pee*, *laissez-faire*)
7. nicknames (such as *Margaret-ka*, *babča-helča*, *mam-ča*)
8. frozen expressions formed by Czech words (such as *více-méně*, *sem-tam*, *jakž-takž*, *vepřo-knedlo-zelo*)
9. URLs
10. tokenization error where more words should be output instead of one (such as *Brno-Praha*, *po-pá* meaning *Monday-Friday*, *voda-vzduch* meaning *water-air*)
11. gender neutral variants[2], such as *chtěl-a bys potkat někoho zajímavého?* (*do you want to meet someone interesting?*), *obráběč-ka kovů* (*machinist*)

[2] see `https://en.wikipedia.org/wiki/Gender_neutrality_in_languages_with_grammatical_gender`

The category 1 should be easily detected by the morphological analyser, categories 2, 3, and 4 could be detected by a proper gazetteer (list of abbreviations, list of proper names, list of chemical names).

Categories 5 and 6 can be recognized by searching corpora for other languages (and we can expect a high number of occurrences of non-English words in English corpora too).

Category 7 is difficult to recognize and it deserves a deep linguistic research.

Category 8 is somewhat similar to categories 5 and 6 but the difference is that occurrences in English corpora will be rare. Similarly to categories 5 and 6, category 8 will often contain words not recognizable by the morphological analyser. Moreover, some of the MWEs can be found in the MWE list available in NLPC[3] created in 2013.

Category 9 is surprisingly not detected by current tools. Nevertheless, dashes in URLs mean often the same thing as spaces between words. Therefore, we can take dashes in URLs into account.

Category 10 will mostly contain words recognizable by the morphological analyser or found in appropriate gazetteer (such as list of place names).

Category 11 could be recognized as a noun or adjective and an ending in the same case. The word without the dash would make no sense.

In further research, we concentrate on categories 5, 6, and 8. The hypotheses are as follows:

- For categories 5 and 6, we could find many occurrences in an English corpus.
- In Czech corpus, we could find enough occurrences of the one word variant (such as *aupair*) and for multiple words variant (such as *au pair*). This applies for all categories 5, 6, and 8 (and probably also for categories 1, 2, 3, and 4).
- Some components of the words in the categories 5 and 8 are subject of inflection.

4.1 Processing the List of Words with Dashes

We filtered out all compound adjectives, i.e. all compound words that are recognizable as a whole by the morphological analyser `majka` or their components are adverb and adjective recognizable by `majka`. In our data, we found 17,777 such occurrences.

We then filtered out all proper names, abbreviations and chemical names found in our gazetteers:

- Czech surnames (provided by Czech ministry of the interior[4]), 8,647 surnames contain a dash
- Czech first names (provided by Czech ministry of the interior[5]), 10,173 names contain a dash

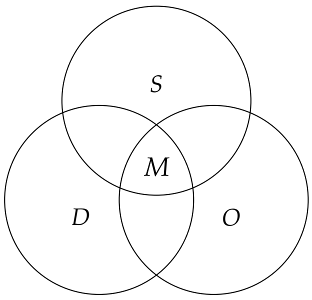
[3] `/corpora/dicts/mwe/xsmerk_mwe.txt`
[4] `http://www.mvcr.cz/clanek/cetnost-jmen-a-prijmeni-722752.aspx`
[5] `http://www.mvcr.cz/clanek/cetnost-jmen-a-prijmeni-722752.aspx`

- place names from Geonames[6], 232,231 names contain a dash
- chemical names from Czech Wikipedia (category Chemistry), 28 words contain a dash
- list of abbreviations provided by Seznam.cz, 82 abbreviations contain a dash

Applying the gazetteers reduced the list of words with dashes to 182,237 occurrences. This step should eliminate many members of categories 1 to 4.

Afterwards, we searched for the variants one word (i.e. after removing the dashes) and the variants with spaces (i.e. replacing dashes with spaces). Figure 4.1 shows how many words were in the intersections of all three sets.

It is necessary to mention that not all occurrences are only orthographic variants of a MWE. For example, we can found *A-Beat* (which is a name of a band), *a beat* (which is either a part of an English phrase or even a part of a Czech phrase).



set	# of occurrences
word with dashes D	182,237
$D \cap$ several words S	65,185
$D \cap$ one word O	46,724
MWE candidates $M = D \cap S \cap O$	26,704

Fig. 1. Number of orthographic variants of MWE candidates.

We identified 26,704 MWE candidates, 5,530 of them contain a one-letter word. MWE candidates with one-letter words can be the most problematic part for disambiguation.

4.2 Selection of Multi-Word Expressions

From the intersection of datasets D, S, and O it can be seen that by far not all words with dashes are good candidates for MWEs.

In this phase, we employed the frequencies computed on the corpus `cztenten12`. For performance reasons, we limited the queries to 1,000.

Let s be the number of occurrences as several words (e.g. *a priori*). Let d be the number of occurrences as one word with dashes (e.g. *a-priori*), let o be the number of occurrences as one word (e.g. *apriori*).We then applied the following decision procedure:

[6] `www.geonames.org`

- if $s > 100 \wedge d > 100$ we found a probable MWE
- if $d < 10 \wedge s > 100o$ we found a normal bigram (no MWE)
- if $d = 1000 \wedge s = 1000$ we found a word with both variants (probably the expression is ambiguous)
- if $s > d \wedge s > o$ and all the conditions above did not apply, we found a MWE

The decision process classified 25,091 expressions, 2,140 of them as MWEs, 332 as probable MWEs, and 174 both MWEs and one word.

5 Results

In Tables 5, we present the most frequent MWEs, probable MWEs, and expressions that can be MWEs and one word at the same time, according to our decision procedure.

It can be seen that still some expressions could be filtered out since they are named entities or noun phrases. We did not employ any language analysis which should be the next step. However, we can see that many MWEs were captured by this procedure.

On the other hand, it seems that the condition the word must occur in all three forms is too strict. Many known MWEs that cause problems with annotation have no occurrence as a word with dashes.

6 Discussion

In some cases, several interpretations of a bigram are possible. For example, *to do* is a strong collocation with English origin that can be found separately in Czech texts. At the same time *to do* are two very common Czech words that can appear in a completely Czech sentence such as *Jsou to do víkendu dva dny* (Two days remaining to the end of the week). Similar case is the fixed expression *po o* meaning *after lunch*, only in the context of kindergartens. One can easily find a Czech sentence where *po o* are two subsequent prepositions.

From these two examples, it can be seen that reckless annotation of all occurrences as frozen MWEs would lead to disaster. This work should therefore be seen as preliminary leading to future re-annotation of the Czech corpus. Next step will consist of context-based filtering of MWE candidates.

7 Conclusion and Future work

In this paper, we present a promising corpus-based tool for recognition of fixed multi-word expressions. The complete list of all expressions including their classification is available at `https://nlp.fi.muni.cz/projekty/mwes/mwe_count.txt`. A web demo that decides whether the input is a MWE or not is available at `https://nlp.fi.muni.cz/projekty/mwes/index.py`.

Table 2. Most frequent MWEs and probable MWEs.

MWEs	probable MWEs	both MWE and one word
reality show	IP adresy	play off
last minute	IP adresu	sem tam
IP adresa	MS Windows	jakž takž
pro rodinné	ne já	open source
i té	PET lahví	Jo jo
IP adres	science fiction	kde kdo
plus mínus	PCI Express	pro aktivní
pole position	in vitro	jo jo
křížem krážem	Sebastian Vettel	ne ne
power play	nahoru dolů	Ne ne
LED diody	SD kartu	pop music
raz dva	Rolls Royce	On Line
shora dolů	in situ	Ski areál
já ty	Pentium M	jel a
MS Word	jakous takous	M ČR
Open Source	Kung Fu	r u
SIM kartu	one man	n i
PS PČR	PEN klubu	po užití
set top	one man show	off topic
LED diod	ready made	P ČR
LCD TV	Obchodní dům	R A
Land Rover	LDL cholesterolu	jisto jistě
MS Excel	No Limit	jakého si

A sound evaluation has not been made yet. We plan to compare lists of MWEs output by our tool with other lists, as well as manual annotation.

As the future work, we plan to implement language analysis in the process, namely morphological analysis and probably simplified syntactic analysis.

Finally, we plan to implement this tool into the annotation pipeline and examine if the corpus annotation improves.

Acknowledgments This work has been partly supported by the Masaryk University within the project *Čeština v jednotě synchronie a diachronie – 2015* (MUNI/A/1165/2014) and by the Ministry of Education of CR within the Czech-Norwegian Research Pogramme in the HaBiT Project 7F14047.

References

1. Bu, F., Zhu, X., Li, M.: Measuring the non-compositionality of multiword expressions. In: Proceedings of the 23rd International Conference on Computational Linguistics. pp. 116–124. COLING '10, Association for Computational Linguistics, Stroudsburg, PA, USA (2010), `http://dl.acm.org/citation.cfm?id=1873781.1873795`

2. Church, K.W., Hanks, P.: Word association norms, mutual information, and lexicography. Comput. Linguist. 16(1), 22–29 (Mar 1990), `http://dl.acm.org/citation.cfm?id=89086.89095`
3. Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing. MIT Press, Cambridge, MA, USA (1999)
4. Michelbacher, L.: Multi-word tokenization for natural language processing. Ph.D. thesis, Universität Stuttgart (2013)
5. Pearce, D.: Synonymy in collocation extraction. In: Proc. of the NAACL 2001 Workshop on WordNet and Other Lexical Resources: Applications, Extensions and Customizations. CMU (2001), `http://www.cogs.susx.ac.uk/users/darrenp/academic/dphil/publications/data/Conferences/naacl2001/paper.pdf`
6. Rychlý, P.: A lexicographer-friendly association score. In: RASLAN 2008. pp. 6–9. Masarykova Univerzita, Brno (2008)
7. Sag, I., Baldwin, T., Bond, F., Copestake, A., Flickinger, D.: Multiword expressions: A pain in the neck for nlp. In: Gelbukh, A. (ed.) Computational Linguistics and Intelligent Text Processing, Lecture Notes in Computer Science, vol. 2276, pp. 1–15. Springer Berlin Heidelberg (2002), `http://dx.doi.org/10.1007/3-540-45715-1_1`
8. Suchomel, V.: Recent czech web corpora. In: Aleš Horák, P.R. (ed.) 6th Workshop on Recent Advances in Slavonic Natural Language Processing. pp. 77–83. Tribun EU, Brno (2012)
9. Šmerk, P.: Unsupervised Learning of Rules for Morphological Disambiguation. In: Sojka, P., Kopeček, I., Pala, K. (eds.) TSD. Lecture Notes in Computer Science, vol. 3206, pp. 211–216. Springer (2004), `http://dblp.uni-trier.de/db/conf/tsd/tsd2004.html#Smerk04`
10. Šmerk, P.: K morfologické desambiguaci češtiny [Towards morphological disambiguation of Czech]. thesis proposal, Masaryk University (2008), `http://is.muni.cz/th/3880/fi_r/`

TIL as Hyperintensional Logic for Natural Language Analysis

Marie Duží[1] and Aleš Horák[2]

[1] VŠB-Technical University of Ostrava
17.listopadu 15, 708 33 Ostrava-Poruba
Czech Republic
`marie.duzi@vsb.cz`

[2] Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno
Czech Republic
`hales@fi.muni.cz`

Abstract. In the paper, we introduce the main ideas of a new scientific project devoted to new methods of computer-aided linguistic and logical analysis of natural language, in particular English and Czech.
The presented project started in 2015 and its main aims lead to the research of the Transparent Intensional Logic (TIL) as a framework for analysis of natural language communication and reasoning. The project follows up the previous successful cooperation between the teams involved and builds new findings on both developed linguistic and logical resources and tools as well as new methods of analysis regarding phenomena such as individual attitudes, dynamic discourse, or tenses and events.

Keywords: Transparent intensional logic; TIL; hyperintensional logic; semantic analysis of natural language

1 Introduction and Related Works

In the area of natural language analysis and knowledge representation much has been done, but even more still needs to be done. Historically, Frege was (to the best of our knowledge) the first to develop a formal semantics. In [1] Frege introduced the well-known semantic schema assigning to expressions their sense (Sinn) and denotation (Bedeutung). Wishing to save compositionality, Frege made the semantics of an expression depend on the linguistic context in which it is embedded. According to Frege an expression names its Bedeutung (extension) in ordinary contexts and Sinn (intension) in oblique contexts. Frege, in an attempt to save compositionality, had recourse to contextualism. The price he paid is too high, though. No expression can, according to Frege, denote an object, unless a particular kind of context is provided. Yet such a solution is far from being natural. There are cases of real ambiguity, witness homonymous expressions. But would anybody say that 'The author of Waverley' were another such a case of homonymy? Hardly. Furthermore, expressions can be embedded within other expressions to various degrees; consider the sentence

"Charles knows that Tom believes that the author of *Waverley* is a poet."

The expression 'The author of *Waverley*' should now denote the 'normal' sense of the 'normal sense' of itself. Adding still further layers of embedding sets off an infinite hierarchy of senses, which is to say that 'The author of *Waverley*' has the potential of being infinitely ambiguous. This seems plain wrong, and is first and foremost an awkward artefact of Fregean semantics (see [8, §1.5]).

The second half of the last century can be characterized as a syntactic turn in semantics. We were developing systems of particular logics which are characterized by a language with a precisely defined syntax and a model set-theoretic semantics. The main goal of building such a system is to find a subset of sentences of the language, axioms of the theory, in fact, which characterize a given area under scrutiny, and then apply proper rules of inference in order to mechanically derive consequences from the axioms. If the system has a model, then it is consistent, and all we are interested in is manipulating symbols. Hence *syntactic turn.*

The mainstream in this direction was *Possible World Semantics* (PWS). Possible-world intensions are extensionally individuated and the PWS semantics is a *logic of intensions*, in particular the *model-theoretic* (hence set-theoretic) *theory of modalities*. Yet its individuation of meaning is too crude (up to logical equivalence only), and thus it is not apt to solve the notoriously well-known problem of the analysis of belief and other attitude sentences. Carnap in [3] says that modal sentences like "It is necessary that *P*" are intensional with respect to the clause *P*. However, sentences about belief like "John believes that *P*" are *neither intensional nor extensional* with respect to *P*. He also criticises Frege's 'naming method' (nowadays we would say 'denotational semantics'), because then we multiply the names *ad infinitum*, and we end up with the antinomy of naming. For Carnap, extensions are not a matter of *logical semantics* because it is a matter of empirical facts and factual knowledge. Prior to the meaning of a term is an *intension* independent of contingent facts that uniquely determines the extension (if any), but not *vice versa*.

In order to solve the problem of belief sentences, Carnap tried to define a stronger relation between expressions than L-equivalence that might rightly calibrate the identity of meaning (i.e. *synonymy*). He defined inductively the relation of *intensional isomorphism* on the set of sentences. Roughly, two sentences *S* and *P* are intensionally isomorphic if they are L-equivalent and each designator (either simple or composed) that is a constituent of *S* is L-equivalent to the respective designator of *P*. Thus sentences *S* and *P* have the same intensional structure if they are composed in the same way from designators with the same intensions. In our opinion, all these tenets and philosophical desiderata of Carnap are plausible and it might seem that he succeeded in analyzing the subjects of beliefs, knowledge, convictions, etc. Moreover, his definition is independent of the language being applied and the syntactic structure in which the clause is encoded. So far, so good; yet Carnap's method was criticized by

Alonzo Church [4]. Church's argument is based on two principles. First, it is Carnap's principle of tolerance (which itself is, of course, desirable), and second, which is less desirable, this principle makes it possible to introduce into a language *syntactically simple* expressions as definitional abbreviations of *semantically complex* expressions. As a result, Carnap's method can yield expressions *P* and *Q* intensionally isomorphic though they obviously have different meanings.

Church proposes *synonymous isomorphism*: all the mutually corresponding designators must be not only L-equivalent but also synonymous, where the synonymy of syntactically simple designators must be *postulated* as a semantic base of a language. We can postulate any convention for introducing these synonymous abbreviations, but as soon as we postulate the meaning of a constant it becomes valid and cannot be changed by another convention. The definition of synonymy occupied Church for many years, which resulted in his Alternatives (0) up to (1). Yet, he was not fully content with any of these proposals.

Since the late 60s of the last century many logicians have strived for *hyperintensional semantics* and *structured meanings* (see, for instance [16]). The structured character of meaning was urged by David Lewis in [23], where non-structured intensions are generated by finite, ordered trees. This idea of 'tree-like' meanings obviously influenced George Bealer's idea of 'intensions of the second kind' in his [1]. The idea of structured meanings was propagated also by M.J. Cresswell who defines structured meanings as ordered *n*-tuples (see [5,6]). That this is far from being a satisfactory solution is shown in Tichý [30], Jespersen [20] and also Bealer [2]. In brief, tuples are set-theoretic entities that are not structured. Besides, tuples are of the wrong making to serve as truth-bearers and objects of attitudes, since a tuple cannot be true or be known, hoped, etc., to be true.

In [25] Moschovakis comes with the idea of *meaning as algorithm*. The meaning of a term *A* is "an (abstract, not necessarily implementable) algorithm which computes the denotation of *A*" ([26, 27]; see also [25]). Yet much earlier, in [27] and [28], Pavel Tichý formulated the idea of *procedural semantics*. Thus, for instance, a sentence encodes an *instruction* how in any possible world at any time to execute the abstract *procedure* expressed by the sentence as its meaning, i.e., to evaluate the truth-conditions of the sentence. He developed a logical framework known today as *Transparent Intensional Logic* (TIL). In modern jargon, TIL belongs to the paradigm of *structured meaning*. However, Tichý does not reduce structure to set-theoretic sequences, as do Kaplan and Cresswell. Nor does Tichý fail to explain how the sense of a molecular term is determined by the senses of its atoms and their syntactic arrangement, as Moschovakis objects to 'structural' approaches in [26, 27].

Tichý's TIL is an overarching logical framework apt for the analysis of all sorts of discourse, whether colloquial, scientific, mathematical or logical. The theory is a *procedural* (as opposed to denotational) one, according to which the meaning of an expression is an abstract, extra-linguistic procedure detailing what operations to apply to what procedural constituents to arrive at the product (if any) of the procedure that is the object denoted by the expression. Such procedures are rigorously defined as TIL *constructions*. TIL proceeds top-down from structured meanings to the entities that these meanings are modes of presentation of. It is a theory that, on the one hand, develops syntax and semantics in tandem while, on the other hand, keeps pragmatics and semantics separate. It disowns *possibilia*; instead the theory operates with a constant domain of individuals for all worlds and times. What vary are the values that (non-constant) intensions have in different worlds and at different times, and not the domains that different worlds and times have. It rejects individual essentialism without quarter, yet subscribes wholeheartedly to intensional essentialism. It denies that the actual and present satisfiers of empirical conditions (possible-world intensions) are ever semantically and logically relevant, and instead replaces the widespread semantic actualism (that the actual of all the possible worlds plays a privileged semantic role) by a thoroughgoing anti-actualism. And most importantly, it unifies unrestricted referential transparency, unrestricted compositionality of sense, and hyperintensional individuation of senses in one theory.

In 2010, the book by Duží, Jespersen and Materna *Procedural Semantics for Hyperintensional Logic* [8] was published in Springer. The book provides an exposition of TIL and its applications as of 2010. Logical semantics is a field progressing by leaps and bounds, and much has happened since Tichý put out his first and only book in 1988 [29]. The 2010 book assembles in one place the most important extensions, improvements and applications stemming from the last several years that address issues not dealt with either at all or only cursorily by Tichý. The book devotes special attention to some topics that generally tend to be dealt with only in passing by contemporary formal semantics. They include, *inter alia*, procedural isomorphism, notional attitudes, knowing whether, concepts (understood rigorously and non-mentalistically), attitudes *de re* and anaphora in hyperintensional contexts. Besides, the extensive treatment of anaphora represents a major step forward for the development of TIL, which had so far barely dealt with this linguistic device. The addition opens up new fragments of natural language to analysis. Another vastly developed notion is *requisite*, which underpins our intensional essentialism (in terms of *a priori* relations-in-extension between intensions). The crown in the jewel is the extremely detailed and principled elaboration of the *de dicto*/*de re* dichotomy. The dichotomy is at the heart of TIL, because it pretty much does the work that is done by reference shift in most other theories.

From the formal point of view, TIL is a hyperintensional, partial, typed lambda calculus. The main feature of lambda calculi is their ability to distinguish between functions and functional values. An additional feature of TIL is its ability to distinguish between functions and *modes of presentation* of functions and their values. We explicate these modes of presentation as abstract *procedures* rigorously defined as TIL *constructions*. Constructions are arranged in a ramified, higher-order type theory that is based on a simple type theory of first-order objects that are non-procedural. The simple type theory, when used for natural-language analysis, spans four ground types (individuals, truth-values, possible worlds, and reals doubling as times) and types of partial functions defined over them. The ramified type theory extends the base of ground types with the types of constructions and types of partial functions defined over them. The typing does not apply to linguistic entities, as in categorial grammar (cf. Montague, Leśniewski, Ajdukiewicz, Cresswell), but to abstract objects such as functions, truth-values, and higher-order entities, as in the constructivist type theory of Martin-Löf. Our bi-dimensional type theory fixes the objective relations among this multi-layered multitude of abstract entities. It thus enables the semanticist to control whether the input is type-theoretically internally coherent and whether the right type of output follows, so as to prevent categorial mismatches.

2 Objectives of the project

This project is interdisciplinary in the sense that the goals we want to achieve concern two closely interrelated areas, viz. computational linguistics and logic.

2.1 Linguistic and logical analysis

The first goal in this area is to make improvements to the *Normal Translation Algorithm*, which is a method that integrates *logical analysis of sentences* with the *linguistic approach* to semantics. The algorithm has been implemented within the previous project. It exploits *complex valency frames (CVFs)* in the *VerbaLex* lexicon of verb valencies (see [17]). The logical analysis module is based on the syntactic analysis result provided by *Synt* module; as a final product, it converts syntactic analyses into formulae in the TIL formalism. To this end we make use of the most important information conveyed by a simple sentence, viz. the verb phrase and its arguments. For the translation of a sentence into a TIL formula, we thus need a wide-coverage lexicon of TIL types assigned to verbs and their arguments. As a result of our intensive work, the TIL type lexicon has been extended up to 10,500 verb-type assignments and about 30 thousands of logical schemata for verbs that serve for assigning correct types to verb arguments thus making it possible to create a proper TIL construction. We make use of VerbaLex lexicon of Czech verb valencies containing deep verb frames. These frames are then used to propose TIL types assigned to verbs and verb logical schemata. In order to assign types to verb arguments, we exploit the links to Princeton WordNet. Finally, the resulting lexicons are manually checked and edited. Currently we have got the corpus of 6,000 TIL constructions that serve for computer-aided analysis of language.

Yet we are still not fully satisfied with the accuracy of the translation. Some sentences are translated into a number of TIL constructions that sometimes differ significantly. Hence we will investigate the causes of these inaccuracies, correct the analyses and erase those that do not match the meaning of a sentence. To this end we must also improve typing in order that it is fully compatible with TIL theory. Here we will make use of the theoretical results in the area of logic and philosophy, in particular of the definition of the three kinds of context, to wit extensional, intensional and hyperintensional, in which the meaning of an expression can occur. The adequacy of the analysis will also be checked by automatically deriving relevant consequences which will be checked manually as for their adequacy. This double checking will yield improvements of the translation algorithm.

The other goal is *bi-lingual analysis*. Here we make use of the definition of procedural isomorphism. Since we explicate structured meanings procedurally, our basic idea is that any two terms or expressions, even in different languages, are synonymous whenever their respective meanings are *procedurally isomorphic*. The notion of procedural isomorphism helps TIL to a principled account of hyperintensional individuation. This is a major issue, because only expressions with procedurally isomorphic meanings are synonymous and can be mutually substituted in hyperintensional contexts.

Yet the synonymy of semantically simple expressions must be established linguistically. To this end we make use of two *very large web corpora*, namely *czTenTen* (for Czech, 5.5 billion tokens) and *enTenTen* (for English, 13 billion tokens). We have designed and developed new tools that are published and publicly used by hundreds of users all over the world – Chared, Onion, JusText and SpiderLing. For an efficient management of such very large corpora, we use the Manatee/Bonito corpus manager developed at the NLP Centre FI MU. Testing on these data showed sufficient speed, coverage and precision of the parsers on general texts from the Internet domain for the Czech language. What remains to be done is transferring the syntactic analysis and logical analysis rules to the English language in order to propose a bi-lingual analysis of general texts in the form of the resulting TIL constructions.

2.2 Logical semantics

First, we plan to improve the analysis of *tenses* as compared to temporal logics, the analysis of *epistemic verbs* and *events*, and the analysis of *ambiguities* in natural language. The foundations of these analyses have been laid down in the previous project No. 401/10/0792 "Temporal Aspects of Knowledge and Information", see [10]. Yet the results deserve to be spelt out further.

Second, we will pursue research on the problem of *synonymy* in natural language. To this end we have defined three variants of *procedural isomorphism* that slightly differ in the degree of individuation of hyperintensions. The first takes only α- and η-equivalent constructions as procedurally isomorphic; the second includes also restricted β-conversion 'by name', and finally the third proposal encompasses α- and β-conversion 'by value' equivalency. Yet we admit that slightly different definitions of procedural isomorphism are still thinkable. What appears to be synonymous in an ordinary vernacular might not be synonymous in a professional language like the language of, for instance, logic, mathematics or physics. Thus we are also considering whether it is philosophically wise to adopt several notions of procedural isomorphism. It is not improbable that several degrees of hyperintensional individuation are called for, depending on which sort of discourse happens to be analysed. Thus the problem of synonymy is still very much an open issue.

Third, we will accomplish the definition of TIL as an extensional logic of hyperintensions (see [9,11,12]). Though TIL's analytical potential is very large, deduction in TIL remains underdeveloped. Tichý defined a sequent calculus for pre-1988 TIL, that is TIL based on the simple theory of types. Since then no other attempt to define a proof calculus for TIL has been presented. The goal is to propose a generalization and adjustment of Tichý's calculus to TIL as per the 2010 book [8]. The adjustments of the calculus concern in particular extensions to the three kinds of context such that it be applicable to hyperintensions within the ramified hierarchy of types. TIL operates with a single procedural semantics for all kinds of logical-semantic context, be it extensional, intensional or hyperintensional. Though operating in a hyperintensional context is far from being technically trivial, it is feasible. To this end we introduce a substitution method that operates on hyperintensions. It makes use of a four-place substitution function (called *Sub*) defined over hyperintensions.

2.3 Communication system

The goal is to apply our analytic methods and application modules so that an interactive *intelligent system* of computer-aided, bi-lingual *communication* is created. To this end we have been developing a computational variant of TIL, viz. the functional programming language *TIL-Script*. The first attempt at a prototype system was accomplished within the five-year (2004–2008) project "Logic and Artificial Intelligence for Multi-agent Systems" that was supported by the Czech Academy of Science. As one of the results of this project, our autonomous intelligent agents can communicate by messaging. The content of messages is encoded in TIL. We developed a small domain ontology of a traffic system both in Czech and English. A noteworthy result was this. Using the common bilingual ontology we could smoothly switch between Czech and English without any programming-code adjustments. Due to hyperintensional features of TIL the agents were able to learn by experience and even recognize ambiguous messages. In such a case the receiver asks the sender for refinement of the message content. The goal of this project is to improve and further develop the system in order to make it compatible with the corpora developed in the NLP Centre FI MU. Moreover, we will improve the analysis of questions so that the system will take into account the fact that questions often come attached with a presupposition. If the presupposition is not valid, the agent replies by a negated presupposition so that the sender can react adequately.

TIL-Script fully complies with TIL, but it is adjusted to the needs of computers. The adjustments concern in particular the syntax of the TIL 'language of constructions'. In TIL-Script we use TeX-like syntax rather than Greek letters, subscripts and superscripts in order to make the language easier to use in computational practice. Moreover, in the interest of better applicability we introduced separate atomic types of real numbers, natural numbers and times, and the types of lists and tuples, though the lists and tuples can be defined as molecular types mapping natural numbers to a particular TIL type. Within the previous projects we implemented syntactic analysis and parsing for TIL-Script and began to build the TIL *inference machine*. The first version of this machine was based on Prolog. In order to extend the calculus to hyperintensional logic of partial functions, we will implement the TIL sequent calculus, which is another goal of this project. At this moment it is an open issue whether we will make use of the general resolution method or implement the calculus directly.

2.4 Summary

In summary, the goals of the project are these.

a) *Logical theory*; further development of TIL, in particular research on
   - the analysis of *tenses*, *presuppositions*, *epistemic verbs*, *events* and *ambiguities* in natural language;
   - procedural isomorphism and the problem of synonymy;
   - TIL sequent calculus
b) *Linguistic and logical analysis*;
   - improvement of the *Normal Translation Algorithm* in order to increase its preciseness and accuracy
   - *bi-lingual analysis* for Czech and English
c) *Communication and agents' attitudes*
   - transformation of a *dialogue* into the *knowledge base*
   - the TIL *inference machine*, the *TIL-Script* functional programming language

3 Methodology and project planning

The project work will run in parallel in three closely cooperating and partly overlapping groups:

- *TIL natural-language processing* group
- *TIL theoretical backgrounds* group
- *TIL inference machine* group

As mentioned above, our theoretical background is Transparent Intensional Logic (TIL) with its procedural semantics. The main tenets of our logical framework are these:

- *Semantic transparency*. We assign TIL constructions to natural language expressions as their context- invariant meanings.
- *Procedural semantics*. TIL constructions are abstract procedures. Thus natural-language expressions encode instructions how, in any state-of-affairs at any time, to evaluate these procedures in order to obtain their product, if any.

Milestones	**Theoretical results**	**Applications**	
		Logical analysis	Inference
1st year (2015)	Study of procedural isomorphism and synonymy; questions and answers with presupposition; logic of dynamic discourse, tenses and events	Computer-aided analysis of individual attitudes in present, future and past tenses and their representation in the knowledge base	Substitution and existential generalization into the three kinds of context while respecting partiality
2nd year (2016)	Resolving ambiguities in natural language; specification of the algorithm of anaphora resolution	Computer-aided analysis of dynamic dialogue based on knowledge bases and ontologies of autonomous agents	Implementation of the algorithm of anaphora resolution in dynamic discourse
3rd year (2017)	Definition of TIL proof calculus; TIL vs. intuitionistic proofs and epsilon calculus	Effective methods of question answering based on knowledge bases and ontologies of autonomous agents	Implementation of the TIL calculus as specified by the theoretical group

Fig. 1. Summary of the project plan.

– *Compositionality*. This principle is closely connected with semantic transparency and Carnap's principle of subject matter. The latter says in principle the following. The constituents of the meaning of an expression can be only constructions of those objects that are explicitly mentioned by the expression. TIL constructions are procedures that spell out how these constituents are unified together into a structured whole.

– *Hyperintensionality*. In TIL ramified theory of types we can easily distinguish three kinds of context, namely extensional, intensional and hyperintensional ones. Yet due to semantic transparency, the meaning of an unambiguous expression is context-invariant. What does depend on a particular context are the objects on which our constructions operate rather than the construction/meaning assignment. In the hyperintensional contexts constructions themselves are objects of predication (though a higher-order construction must be used to produce this lower-order argument construction). In the intensional contexts the products of constructions, that is set-theoretical functions, are the objects of predication. And in the exten-

sional contexts functional values are the objects of predication that get operated on.
- *Extensional logic of hyperintensions*. The above principles altogether make it possible to develop an extensional calculus of hyperintensions in which the extensional principles like existential generalisation and substitution of identicals are valid.

Logical analysis of natural language will adhere to these principals. Moreover, we aim to integrate computer-aided linguistic analysis with logical analysis. In particular, we must still improve the products of linguistic typing in order to match them with logical typing. We will improve the Normal Translation Algorithm, a method that integrates *logical analysis of sentences* with the *linguistic approach* to semantics. The algorithm exploits *complex valency frames* in the VerbaLex lexicon of verb valencies.

Concerning the development of TIL inference machine, much has been done yet still more remains to be done. The results have been presented at respectable international conferences and published in journals. What remains is the specification of the rigorous extensional calculus of hyperintensions and its implementation in the computational variant of TIL, viz. the *TIL-Script* language.

4 Conclusions

We have presented the main ideas of a just started research project that aims to built on the logical framework of the Transparent Intensional Logic (TIL) as a basis for complex analysis of higher-order semantic phenomena of natural languages. As a subject study, the project will verify all findings on at least two representatives of a range of natural languages, namely Czech, as a morphology-rich free-word-order language, and English, as a mainstream example of analytical language.

We believe that fulfilling the project goals will move the research in natural language processing beyond the well-known shortcomings of set-theoretical natural-language semantics. Moreover, computer-aided analysis of natural language will make it possible to develop an intelligent system of communication both in Czech and English.

Acknowledgements This work has been partly supported by the Ministry of Education of CR within the LINDAT-Clarin project LM2010013 and 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. Bealer, G. (1982): *Quality and concept*. Oxford: Clarendon Press.

2. Bealer, G. (2004): An inconsistency in direct reference theory. *Journal of Philosophy* 101: 574–593.
3. Carnap, R. (1947): *Meaning and necessity*. Chicago: Chicago University Press.
4. Church, A. (1954): Intensional isomorphism and identity of belief. *Philosophical Studies* 5: 65–73.
5. Cresswell, M.J. (1975): Hyperintensional logic. *Studia Logica* 34: 25–38.
6. Cresswell, M.J. (1985): *Structured meanings*. Cambridge: MIT Press.
7. Duží M. (2008): TIL as the Logic of Communication in a Multi-Agent System. *Research in Computing Science*, 2008, vol. 33, special issue Advances in Natural Language Processing and Applications, pp. 27–40. ISSN 1870-4069.
8. Duží M., Jespersen B. and Materna P. (2010): *Procedural Semantics for Hyperintensional Logic. Foundations and Applications of Transparent Intensional Logic*. First edition. Berlin: Springer, series Logic, Epistemology, and the Unity of Science, vol. 17, ISBN 978-90-481-8811-6, 550 pp.
9. Duží, M. (2012a): Extensional logic of hyperintensions, in *Conceptual Modelling and its Theoretical Foundations*, A. Duesterhoeft, M. Klettke and K.-D. Schewe (eds.), LNCS 7260, Berlin-Heidelberg: Springer 2012, pp. 268–290, ISBN 978-3-642-28278-9.
10. Duží, M. (2012b): Resolving topic-focus ambiguities in natural language. In *Semantics in Action – Applications and Scenarios*. Ed. Muhammad Tanvir Afzal, Croatia: InTech Europe, 2012, pp. 239–266, ISBN 978-953-51-0536-7.
11. Duží, M. (2012c): Towards an extensional calculus of hyperintensions. *Organon F*, Vol. 19, supplementary issue 1, pp. 20–45.
12. Duží, M. (2013): Deduction in TIL: From simple to ramified hierarchy of types. *Organon F*, vol. 20, supplementary issue 2, pp. 5–36.
13. Duží, M., Jespersen, B. (2012): Transparent quantification into hyperpropositional contexts *de re*. *Logique and Analyse*, vol. 220, pp. 513–554.
14. Duží, M., Jespersen, B. (2013): Procedural isomorphism, analytic information and beta-conversion by value. *Logic Journal of the IGPL*, vol. 21, pp. 291–308.
15. Fellbaum, Ch. (1998, ed.): *WordNet: An Electronic Lexical Database*. Cambridge, MA: MIT Press.
16. Fox, Ch. and Lappin, S. (2010): Expressiveness and complexity in underspecified semantics, *Linguistic Analysis*, vol.36.
17. Hlaváčková, D., Horák, A. (2006): VerbaLex - New Comprehensive Lexicon of Verb Valencies for Czech. In: *Computer Treatment of Slavic and East European Languages*. Bratislava, Slovakia: Slovenský národný korpus, 2006. pp. 107–115.
18. Horák, A. (2008): *Computer Processing of Czech Syntax and Semantics*. Brno, Czech Republic: Librix.eu, 241 pp.
19. Horák, A., Pala, K., Materna, P., Duží, M. (2007): Verb Valency Semantic Representation for Deep Linguistic Processing. In *ACL 2007, Proceedings of the Workshop on Deep Linguistic Processing*. Praha: The Association for Computational Linguistics (ACL), 2007. pp. 97–104.
20. Jespersen, B. (2003): Significant sententialism in Transparent Intensional Logic and Martin-Löf's type theory, in: *The Logica Yearbook 2002*, T. Childers, O. Majer (eds.), Czech Academy of Sciences, Prague (2003), 117–31.
21. Jespersen, B. and G. Primiero (2010): Two kinds of procedural semantics for privative modification, *Lecture Notes in Artificial Intelligence*, vol. 6284 (2010), 252–271.
22. Jespersen, B. and G. Primiero (2013): Alleged assassins: realist and constructivist semantics for modal modification. *Lecture Notes in Computer Science*, vol. 7758 (2013), 94–114 (Revised Selected Papers).

23. Lewis, D. (1972): General semantics. In *Semantics of Natural Language*, eds. D. Davidson and G. Harman, 169–218. Dordrecht: Reidel.
24. Martin-Löf, P. (1984): *Intuitionistic Type Theory*. Bibliopolis, Naples.
25. Moschovakis, Y.N. (1994): Sense and denotation as algorithm and value. In *Lecture Notes in Logic*, eds. J. Väänänen and J. Oikkonen, vol. 2, 210–249. Berlin: Springer.
26. Moschovakis, Y.N. (2006): A logical calculus of meaning and synonymy. *Linguistics and Philosophy* 29: 27–89.
27. Tichý, P. (1968): Smysl a procedura. *Filosofický časopis* 16: 222–232. Translated as 'Sense and procedure' in [31, 77–92].
28. Tichý, P. (1969): Intensions in terms of Turing machines. *Studia Logica* 26: 7–25. Reprinted in [31, 93–109].
29. Tichý, P. (1988): *The Foundations of Frege's Logic*. Berlin, New York: De Gruyter.
30. Tichý, P. (1994): The analysis of natural language. *From the logical point of view* 3: 42–80. Reprinted in [31, 801–841].
31. 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.

Generating Czech Iambic Verse

Zuzana Nevěřilová and Karel Pala

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

Abstract. In the paper, we describe an algorithm for generating Czech iambic verse and its implementation on a computer. It is a continuation of the work first done in 1972 [6], in which a program generating Czech iambic verse had been developed, written in the programming language Algol-Genius and run on the mainframe SAAB D21 with interesting results.
Here, we present a new experiment which is a follow-up of the previous one and includes some new techniques as e.g. using a complete Czech automatic morphology (Majka) plus small DC grammar of Czech. The poetic rules built before 1972 by Jiří Levý are used with only slight modifications. SWI Prolog has been selected as a programming language. Also the resources are different, as a base for the poetic vocabulary we have decided to use the literary texts from Czech Wikisources.

Keywords: poetry, natural language generation, definite clause grammars, semantics

1 Introduction

The goal of the paper is to model on a computer the creative processes, in which Czech iambic verse is produced. We describe an algorithm for generating Czech iambic verses and the implementation of the whole procedure. In other words, we are interested in some general aspects of computer modeling of some human brain functions.

A computer generating of Czech verse was initiated by J. Levý in 1966 who outlined the three main objectives:

– Formulating the poetic rules which would enable us to generate Czech verses with various metrum on the computer: the first attempt was made for the ten-feet iambic verse by J. Levý shortly before he died (on January 1967, [4]).
– Building the proposed poetic rules into a formal (context-free) grammar of Czech and implementing it as a program generating Czech sentences after confrontation of grammar and poetic principles.
– Analysing and evaluating the obtained results both syntactically and semantically and comparing them with the human creations.

2 Related Works

According to our knowledge, not much attention had been paid to this sort of research in Czech context, thus it is difficult to offer a methodological as well as technical comparison of solving similar problems. The text by J. Novotný[1] is a general and philosophical talk which does not contribute much to the considered area.

An attempt to deal with the Czech rhymes was created by P. Šrubař[2], it is interesting, however, it does not touch the grammar and the rules for a metre. There is an interesting text by J. Materna[3] which informs about applying machine learning techniques to this issue. We also would like to mention the text *Poetry, computer and poet* [9] which contains an overview of the experiments in the field.

Other projects concerning generation of poems include the famous Raymond Queneau 100,000,000,000,000 sonnets, or many attempts that generate poems from patterns (e.g. AI poems[4]). Works similar to ours are rather rare but exist, e.g. Automatic Poetry Generator[5].

3 Lexicon: Corpus of Czech Poetry

Compared to the early work [4], we have immense possibilities of computational power and storage. In the early work, the authors created the lexicon from some writings by Jaroslav Seifert. Now we have decided to create the lexicon automatically. We created a 2 million corpus of Czech poetry (and Czech translations of non-Czech poetry) from Wikisources[6]. The corpus has been annotated using tokenizer `unitok`, and the tagger `desamb` [5]. Afterwards, we let the Sketch Engine [3] compute word sketches and exported the most frequent 1000 words.

It can be seen that the language of poems differs significantly from the language in general corpora. Among the 1000 most frequent words from the poetry corpus and the big web corpus `cztenten12`, only 423 word were in both corpora. To compare, the same comparison between `cztenten12` and another Czech corpus `Czes2` founds 817 common words.

"A word sketch is a one-page, automatic, corpus-derived summary of a word's grammatical and collocational behaviour" [1]. In the poetry corpus, the word sketches differ a lot compared to the `cztenten12` corpus. Not surprisingly, the distributional thesaurus for the poetry corpus differs from the

[1] `http://www.fi.muni.cz/usr/jkucera/pv109/2004/51914-pocitace_a_poezie.html`

[2] `http://www.rymy.cz/about.htm`

[3] `http://www.mlguru.cz/stredovek-umele-inteligence-skoncil-seznamte-se-s-neuronovymi-sitemi-ktere-umi-psat-basne/`

[4] `http://www.aipoem.com`

[5] `http://www.languageisavirus.com/automatic_poetry_generator.html`

[6] `https://cs.wikisource.org`

thesaurus in `cztenten12`. We benefit from the thesaurus when building a stanza from the verses (see Section 5).

4 Formal (DC) Grammar Generating Czech Sentences

Similarly to the previous work, we focused on Czech iambic verse. In Czech poetry, trochees dominate, iambic metre is somewhat unusual because the stress is always on the first syllable. Thus, Czech iambic verses have to start with monosyllabic words but in Czech, most words are longer than one syllable. Nevertheless, many Czech poets wrote iambic poetry or combined verses (e.g. alternation of 5-feet and 4-feet iambic verse or alternation of iambic and trochaic verse). In order to generate Czech verses with a given metre we have to deal with Czech grammar, verb valencies, and poetic rules.

4.1 Grammar for Czech Sentences

The language structure, i.e. grammar: in our case, we decided to work with a small definite clause grammar (DCG) [7] of Czech named Klara and developed in the NLP Centre. It is a formal device able to recognize (and generate) Czech sentences. We adapted a subset of Klara grammar in order to generate correct Czech sentences. The subset concerns noun phrases (with adverbial and adjective modifiers), pronominal phrases, verb phrases (incl. compound verbs), and simple adjective phrases.

In Czech we have to deal with two types of obligatory grammatical agreement that are satisfied in the grammar:

– grammatical agreement in gender, number and case among all components of a noun or pronominal phrase, and between the subject
– grammatical agreement in gender and number between the subject and the verb in past tense or conditional

We also included interjections and particles. First, they appear in poetry (probably more often than in other texts), second, many interjections and particles are monosyllabic words and therefore are useful as first words in iambic verses (see Section 4.3). Using the grammar outlined in this Section, we generated over 1,8 millions of verses.

4.2 Verb Valencies

The rules of our DC grammar also allow us to capture partly semantic aspects of the generated sentences using valency frames (VerbaLex [2]). In this way, we may control the meaningfullnes of the generated sentences in a reasonable extent. DCG produces not only Czech sentences but also their syntactic representations in the form of syntactic trees. In the preceding experiment with generating Czech verses, a fragmentary context-free grammar was used [6]. This time, we have decided to work with a DC grammar which is more adequate for handling natural languages (like Czech). We extracted right verb valencies (corresponding to the object) from all verbs in the lexicon using VerbaLex. Although a verb can have several right valencies, we took only the first for each verb synset. For this reason, our system cannot generate sentences such as *Peter went from Prague to Brno* but only *Peter went from Prague* and *Peter went to Brno*.

4.3 Poetic rules for the Czech iambic verse

The poetic rules working on the language material, i.e. on the Czech sentences produced by the DC grammar mentioned above were implemented directly as a DCG constraint. In the previous work [6], the poetic rules were applied to generated phrases.

First, we have to define some basic notions. A syllable is a group of letters containing one vowel or diphthong or one syllabic consonant, i.e. r or l in the environment `CrC` or `ClC`. A word with $1 \dots n$ syllables is a word containing $1 \dots n$ vowels. The Czech vowels are the following: a, á, o, ó, u, ů, ú, i, í, e, é, y, ý, ou, au, eu plus the syllabic consonants r, l (also m, n could be considered but we leave them out here).

Each syllable (vowel) in a verse is considered to be a position (symbollicaly denoted by p, each position is either even or odd. A null position, i. e. a position immediately before the first occupied position, is automatically treated as an even position. For generating an N-syllable iamb (a $N/2$-feet iambic verse), we assume that we have N occupiable positions.

The following rules of word selection and rules regulating the number of syllables can be formulated:

1. If the last occupied position is even, i. e. if $p = 0, 2, 4, \dots$, it is necessary to select a monosyllabic word. This means that each verse has to start with a monosyllable.
2. If the last occupied position is odd, i. e. $p = 1, 3, 5, \dots$, it is necessary to select a $2 \dots n$-syllabic word whose maximum number of syllables follows from the relation $n = N - p$ (without enjambments $n = N - 1 - p$).

2b. A weaker formulation: if the last occupied position is odd, i. e. if $p = 1, 3, 5, \dots$, it is possible to select any word X, even a monosyllable, whose maximal number of syllables follows from the relation $n = N - p$ ($n = N - 1 - p$). This rule, if used, would cause a considerable loosening of the rather rigid poetic rules.

Rules for rhymes such as `aa`, `bb`, `cc` ... or `ab ab cd cd` ... or `abba`, `cddc` ... in a iambic verse can formulated as well [6] but we plan to apply them in the future research.

In contrast with the previous work, we implemented the poetic rules directly to the DC grammar. In order to generate mostly sentences that correspond to the poetic rules, we start each sentence with a monosyllabic word and afterwards, we generate n-syllable iambs for $n \in 5, 8, 9, 10, 11, 12$. The verses have to have stress at least on some even syllables, so the program generates words longer than two syllables. This approach is close to the rule 2b.

In the second stage, we combine the generated verses in order to create a stanza. We observed the stanzas in poems by Czech poets Boleslav Jablonský (19th century, early romantism), Karel Sabina (19th century, romantism), and František Hrubín (20th century, modernist) and copied their schemes. For example in Hrubín's poem Milostná, the stanza contains verses of 8, 11, 5, 8, 11, 5 syllables, most of them purely iambic.

5 Adding Semantics to the Generated Verse

The combination of generated verses is partially random but constrained by the poetic rules for Czech iambic verses and also by the patterns of stanzas. These constraints can lead into non-sense combination such as illustrated in Example 5. No semantic relation among the words větvím (branches), mořím (seas), hudbám (music), hadovi (snake) is apparent. For this reason, we employed the Sketch Engine Thesaurus [8] in order to keep a particular topic for most of the verses in the stanza. Such interconnected verses that form a stanza are shown as Example 5.

To conclude, the generated verses are constrained not only by grammatical (agreements), syntactical (word order, verb valencies), and poetical (number of syllables, stress) rules but also partially by semantic rules. For this reason, we need a very large number of verses from which the program can select the appropriate ones.

co bere celým starým větvím kde
jen dám si novým mořím
co bere celým bílým hudbám ven
sem bijeme hadovi

Fig. 1. A stanza without semantic information.

ty výš běžela do kruhů
co pryč bijeme **obrazům**
kde dám **kráse**
co kdes bijeme **otcovi**
co hodně bijeme věci
též dám **krásám**

Fig. 2. A stanza with the topic víra (belief), words similar to the topic are marked in bold.

Sketch Engine Thesaurus for víra (belief) : **obraz** (**image**), pravda (truth), církev (the Church), vůle (will), domov (home), myšlénka (thought), právo (right), milost (compassion), cit (feeling), rozkoš (delight), místo (place), hřích (sin), svoboda (freedom), štěstí (happiness), vlast (homeland), řád (order), mír (peace), vděk (gratitude), řeč (speech), víska (small village), naděje (hope), radost (joy), ... **krása** (**beauty**), nevěsta (bride), čest (honor), **otec** (**father**) ...

6 Conclusions and Future Work

We have built a Prolog DC grammar that generates Czech iambic verses using a thesaurus-based stanza builder. The program can be used via the web interface at `https://nlp.fi.muni.cz/projekty/czech_verse/`. The quality of the resulting poetry is going to be a subject of a future poetry evaluation in a cooperation with the versologists.

So far, we have not implement rhymes in the stanza builder. This feature will be developed in the future.

In the paper, we have been trying to model a situation, in which a poet – human being – selects some syntactic structures and refuses other because of the inconsistence with his poetic intentions. The analogy between the poet-creator and a computer is rather superficial – the computer's creative intentions are dependent on our ability to formulate as adequately as possible the creative properties of a poet. We have to realize that the machine is able to solve the conflicts using combinatoric techniques only – by the systematic search of all existing variants until the first acceptable one is found.

One important remark has to be made: in the paper, we have not paid much attention to the semantic aspects of the poetic creation though they play a relevant role in this respect. The semantics influences strongly a poetic message that poets try to express in their verses. Not speaking about integrating emotions into poetic creations. In this respect, our possibilities are quite limited so far even though an existing research in the area of the affective computing offers some progress. But this is a topic for another paper.

Acknowledgement This work has been partly supported by the Masaryk University within the project *Čeština v jednotě synchronie a diachronie – 2015* (MUNI/A/1165/2014) and by the Ministry of Education of CR within the Czech-Norwegian Research Pogramme in the HaBiT Project 7F14047 and within the LINDAT-Clarin project LM2010013.

References

1. Word sketch | Sketch Engine, `https://www.sketchengine.co.uk/word-sketch/`, [accessed online, 2015-11-01]
2. Hlaváčková, D., Horák, A.: Verbalex – new comprehensive lexicon of verb valencies for Czech. In: Proceedings of the Slovko Conference (2005)
3. Kilgarriff, A., Rychlý, P., Smrž, P., Tugwell, D.: The Sketch Engine. In: Proceedings of the Eleventh EURALEX International Congress. p. 105–116 (2004), `http://www.fit.vutbr.cz/research/view_pub.php?id=7703`

4. Levý, Jiří, Pala, K.: Generování veršů jako problém prozodický [Generating verses as a prosodic problem]. In: Palas, K.; Levý, J. (ed.) Teorie verše. II, Sborník druhé brněnské versologické konference, 1967
5. Šmerk, P.: K morfologické desambiguaci češtiny [Towards morphological disambiguation of Czech]. thesis proposal, Masaryk University (2008), `http://is.muni.cz/th/3880/fi_r/`
6. Pala, K.: Conflicts between grammar and poetics. In: Prague Studies in Mathematical Linguistics IV. pp. 229–240. Czechoslovak Academy of Sciences (1973)
7. Pereira, F.C.N., Warren, D.H.D.: Definite Clause Grammars for Language Analysis - A Survey of the Formalism and a Comparison with Augmented Transition Network. Artificial Intelligence pp. 231–278 (1980)
8. 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. pp. 41–44. ACL '07, Association for Computational Linguistics, Stroudsburg, PA, USA (2007), `http://dl.acm.org/citation.cfm?id=1557769.1557783`
9. Uličný, O.: Poezie, počítač a básník [Poetry, computer and poet]. Čeština doma a ve světě 5(3), 27–32 (1997)

Examples of computer generated poetry (Czech)

ty tady běžela do věr
co bere nebesům místo
tak sem chvěje mojí krásou
co ráno bijeme hříchu

on, jenž byl svět, byl co milejší ret
co bere celým krásným rakvím též
on, jenž byl svět, byl ty dobrý věk
co bere celým novým slastem ráz
tak tak čekají jeho malého
on, jenž byl svět, byl on tichý bez

tak též cítila se neplné časy
ty hlouběji běžela do časů
co nic bijeme celým bílým zářím
tak tu cítila se nevětší ženu
tak ještě cítila se plný den
on, jenž dal srdci, dal si drahým stromům
tak nedaleko čekají mé rety
dnes bijeme hradům

co bere celým jiným věžím nic
ty nejlépe běžela do ramen
tak hodně chvěje jeho horou
co bere celým zlatým dveřím nic
ty někdy běžela do matičky

tak radší chvěje její horou

on, jenž byl svět, byl on velký les
co bere celým velkým stráním hoře
tak radši cítila se mladší jiné
co bere celým starým hudbám klidně
co bere celým zlatým růžím sladce
on, jenž dal srdci, dal nám mladým jiným

Author Index

RASLAN 2015

Ninth 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 2015

ISBN 978-80-263-0974-1
ISSN 2336-4289