RASLAN 2012
Recent Advances in Slavonic Natural Language Processing

A. Horák, P. Rychlý (Eds.)

RASLAN 2012

Recent Advances in Slavonic Natural Language Processing

Sixth Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2012
Karlova Studánka, Czech Republic, December 7–9, 2012
Proceedings

Tribun EU
2012

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

ISBN 978-80-263-0313-8

Preface

This volume contains the Proceedings of the Sixth Workshop on Recent Advances in Slavonic Natural Language Processing (RASLAN 2012) held on December 7th–9th 2012 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 discussions about new ideas. The intention is to have it as a forum for presentation and discussion of the latest developments in the field of language engineering, especially for undergraduates and postgraduates affiliated to the NLP Centre at FI MU. We also have to mention the cooperation with the Dept. of Computer Science FEI, VŠB Technical University Ostrava.

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 2012 offers a rich program of presentations, short talks, technical papers and mainly discussions. A total of 16 papers were accepted, contributed altogether by 24 authors. Our thanks go to the Program Committee members and we would also like to express our appreciation to all the members of the Organizing Committee for their tireless efforts in organizing the Workshop and ensuring its smooth running. In particular, we would like to mention the work of Aleš Horák, Pavel Rychlý and Lucia Kocincová. The TEXpertise of Adam Rambousek (based on LATEX macros prepared by Petr Sojka) resulted in the extremely speedy and efficient production of the volume which you are now holding in your hands. Last but not least, the cooperation of Tribun EU as both publisher and printer of these proceedings is gratefully acknowledged.

Brno, December 2012 Karel Pala

Table of Contents

IV Language Modelling

Part I

Syntax, Morphology and Lexicon

Saara: Anaphora Resolution on Free Text in Czech

Vašek Němčík

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

Abstract. Anaphora resolution is one of the key parts of modern NLP systems, and not addressing it usually means a notable performance drop. Despite the abundance of theoretical studies published in the previous decades, real systems for resolving anaphora are rather rare. In this article we present, to our knowledge, the first practical anaphora resolution system applicable to Czech free text. We describe the individual stages of the processing pipeline and sketch the data format used as an interface between individual modules.

Key words: anaphora resolution, Saara, Czech

1 Introduction

In this work, we present a natural language processing (NLP) application setting capable of anaphora resolution (AR) based on plain free text in Czech. This is accomplished by combining several NLP tools, described below, developed at the NLP Centre at the Masaryk University in Brno.

When analyzing texts, anaphoric expressions, especially pronouns, require special handling. On their own, they do not contain any semantic information, and therefore traditional morphological tagging or syntactic analysis as such do not make it possible to arrive at their full meaning. To obtain a complete representation of sentences containing pronouns, these need to be considered in context, namely, interpreted by an AR procedure. Failing to incorporate such a procedure into an NLP system means accepting only a partial text representation, and often a subsequent performance drop.

To our knowledge, there is only a limited number of stand-alone AR systems that work with plain text input, and we are not aware of any such system available for Czech.

In the next section, we mention similar anaphora resolution systems proposed so far. Section 3 describes the processing pipeline of Saara, and further, Section 4 presents performance figures. Finally, we sketch directions of our further research.

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2012, pp. 3–9, 2012.

2 Related Work

This section sums up existing systems relevant from our perspective, starting with complex AR systems for English, followed by proposals made for Czech.

A number of software tools for performing AR have been presented in the recent years. One of the prominent ones is MARS (Mitkov, Evans, and Orăsan, 2002), a system created at the Univeristy of Wolverhampton. The core of the underlying AR algorithm is a weighting scheme based on the so-called antecedent indicators. There are versions of MARS for various languages, such as English, French, Arabic, Polish, or Bulgarian.

A further notable AR system is BART (Versley et al., 2008), a product of inter-institute cooperation encouraged by the Johns Hopkins Summer Workshop in 2007. BART is a framework allowing straightforward experimenting with various machine learning models. It operates over XML data and allows easy visualisation of results in the MMAX tool (Müller and Strube, 2006).

For Czech, mainly theoretical work has been published. First theoretical models have emerged from the long tradition of research on the Functional Generative Description (FGD) of language. Several algorithms were proposed, for instance by Hajičová (1987), Hajičová, Hoskovec, and Sgall (1995), and Hajičová, Kuboň, and Kuboň (1990), providing only tentative evaluation, due to the lack of sufficiently large annotated data at that time.

The emergence of the Prague Dependency Treebank (PDT) (Hajič et al., 2005), a richly annotated Czech treebank containing annotation of pronominal anaphora, made it possible to experiment with AR systems and to evaluate them. Apart from our work, a notable AR system for Czech is AČA presented by Linh (2006). It comprises rule-based algorithms and also machine learning models for resolving individual pronoun types. Further, a noun phrase coreference resolution system based on maximum entropy and perceptron models was proposed by Novák and Žabokrtský (2011). These systems are respectable results in the field of Czech computational linguistics, however, are fitted to the dependency-based formalism of PDT and their applicability to data in other formalisms may be limited.

The next section gives more details about Saara, a stand-alone AR system for Czech.

3 Saara Pipeline

Saara is is a modular AR system, currently containing re-implementations and variants of selected salience-based algorithms. The architecture of the system was inspired by the principles suggested by Byron and Tetreault (1999), the key points being modularity and encapsulation. They suggest segmenting system modules into three layers: Themselves, they propose three layers:

- the translation layer for creating data structures,
- the AR layer containing functions addressing AR itself,
- the supervisor layer for controlling the previous layers.

```
<s id="sent1" class_id="cls1" type="sentence">
<markable id="m1" class_id="cls2" type="clause">
<markable id="m2" class_id="cls3" type="np" gram="subj">
Filip          Filip          k1gMnSc1
</markable>
políbil        políbit        k5eAaPmAgInSrD,k5eAaPmAgMnSrD
<markable id="m3" class_id="cls4" refconstr="m2" type="np" gram="obj">
Lucii          Lucie          k1gFnSc4
</markable>
</markable>
<g/>
.              .              kIx.
</s>
<s id="sent2" class_id="cls5" type="sentence">
<markable id="m4" class_id="cls6" type="clause">
<markable id="ms1" anaref="m2" class_id="cls3" type="pron_pers_zero" gram="subj">
_              on             k3p3gMnSc1,k3p3gInSc1,k3p3gFnSc1,k3p3gNnSc1
</markable>
Miluje milovat k5eAaImIp3nS
<markable id="m5" anaref="m3" class_id="cls4" refconstr="ms1" type="pron_pers_weak" gram="obj">
ji             on             k3p3gFnSc4xP
</markable>
</markable>
<g/>
.              .              kIx.
</s>
```

Fig. 1. An example of a structured vertical file

We adopt an anologous scheme of layers: the *technical layer* of scripts converting data from various formalisms into a general linear format containing structural tags, so-called markables and their attributes; the *markable layer* abstracting from formalism specifics, operating solely over the already known markables and their attributes, and focusing on the AR process as such; and finally the *supervisor layer* defining the application context, such as individual pre-processing steps and AR algorithm settings.

The interface between all modules is the so-called *structured vertical file*, a plain text format containing one line per token, with extra tags expressing higher-level units, such as sentences, clauses and referential expressions. A slightly abridged example of such a file is given in Figure 1.

The first phase of the processing is converting the input data into the vertical format and performing **morphological analysis**. For plain text input, this is performed by desamb (Šmerk, 2007), a Czech tagger assigning morphological tags to each token and disambiguating these tags based on a statistical model and a set of heuristic rules. For words that can not be disambiguated based on shallow linguistic information, such as pronouns, multiple morphological tags are restored by the Majka morphological analyzer (Šmerk, 2009). At the end of this phase, each token line contains a morphological tag and lemma.

Next, **syntactic analysis** is performed using either the SET (Kovář, Horák, and Jakubíček, 2011) or Synt parser (Jakubíček, Horák, and Kovář, 2009). We use the SET parser by default, as it is slightly more robust. It is based on a small set of rules detecting important patterns in Czech text. In the Saara pipeline,

Table 1. Performance of the system in MUC-6 and traditional measures

	MUC-6		IR-style	
	Precision	Recall	Precision	Recall
Plain Recency (baseline)	22.40	24.85	20.75	20.75
BFP Centering	42.36	44.85	38.98	37.47
Lappin and Leass' RAP	36.54	40.41	35.49	35.39

we use SET to extract phrases, which are subsequently incorporated into the vertical file as tags grouping tokens in question.

As the next step, necessary **syntactic post-processing** is carried out. This comprises assignment of coarse-grained grammatical roles, and based on that, detection of zero subjects, which are afterwards re-constructed as dummy tokens and makrables, including their morphological features.

The core phase of the computation is **anaphora resolution** as such. Modules implementing viriations of diverse AR algorithms, such as the BFP algorithm (Brennan, Friedman, and Pollard, 1987) or RAP (Lappin and Leass, 1994), are available. AR modules supplement markable tags representing individual discourse objects with information about their antecedents and coreference classes.

A web version of this application setting, accepting Czech free text, and with Saara configured to resolve personal pronouns, is freely available online at `http://nlp.fi.muni.cz/projekty/anaphora_resolution/saara/demo/`. We hope the availability of this demo will encourage experimenting with Saara and its extrinsic comparison with other systems.

4 Evaluation

Evaluation of AR systems (and complex NLP systems in general) is a rather complicated issue and gives rise to frequent misconceptions.

A number of sophisticated metrics have been proposed to assess the performance of AR systems with precise numbers, however, these numbers are often substantially biased by a broad range of factors not pointed out in the evaluation report. The figures largely depend on

- whether the evaluation is performed on manually corrected data or data susceptible to processing errors,
- whether errors propagated from the pre-processing (ie. tagging, markable detection) are counted,
- whether all errors are counted equally,
- the precise types of anaphora addressed,
- the size and genre of the text etc.

To evaluate Saara, we use the PDT data projected into structured verticals by the pdt2vert tool (Němčík, 2011), considering only personal pronouns, namely strong and weak personal pronouns, and zero subjects of finite verb groups (the total of 8648 anaphors). We are aware of the fact that the data may contain errors, for instance, due to imperfect detection of clause boundaries, however, we adopt the given structures as correct. Anaphors resolved to a different member of the same coreference chain are considered to be resolved correctly, and all errors have the same weight.

To compare the individual algorithm prototypes, their performance is revealed in Table 1. These results need to be considered as tentative, and are expected to improve with further parameter tuning and the contribution of anaphoric links of further types.

5 Future Work

We have described Saara as a part of a stand-alone NLP system accepting plain text as input, and performing syntax analysis supplemented by an interpretation of personal pronouns.

In our future work, we mainly aim at enhancing the AR methods by accounting for further information relevant to the antecedent choice. The long-term goal is to incorporate as much semantic information as possible with respect to its availability and the reliability of the lower-level analysis. As a first step, a decent approximation can be obtained by considering word co-occurrence statistics.

Further, we plan on to account for further types of anaphoric expressions, for example, certain uses of demonstrative pronouns. Demonstrative pronouns are rather complex to resolve, as they allow reference to abstract entities and discourse segments of arbitrary size.

Acknowledgments

This work has been partly supported by the Ministry of Education of the Czech Republic project No. LM2010013 (Lindat–Clarin – Centre for Language Research Infrastructure in the Czech Republic).

References

1. Brennan, Susan E., Marilyn W. Friedman, and Carl J. Pollard. 1987. A centering approach to pronouns. In *Proceedings of the 25th Annual Meeting of the ACL*, pages 155–162, Standford.
2. Byron, Donna K. and Joel R. Tetreault. 1999. A flexible architecture for reference resolution. In *Proceedings of the Ninth Conference of the European Chapter of the Association for Computational Linguistics (EACL-99)*.

3. Hajič, Jan et al. 2005. The prague dependency treebank 2.0, `http://ufal.mff.cuni.cz/pdt2.0/`. Developed at the Institute of Formal and Applied Linguistics, Charles University in Prague. Released by Linguistic Data Consortium in 2006.
4. Hajičová, Eva. 1987. Focusing – a meeting point of linguistics and artificial intelligence. In P. Jorrand and V. Sgurev (eds.), *Artificial Intelligence vol II: Methodology, Systems, Applications*. Elsevier Science Publishers, Amsterdam, pp 311–321.
5. Hajičová, Eva, Tomáš Hoskovec, and Petr Sgall. 1995. Discourse modelling based on hierarchy of salience. *The Prague Bulletin of Mathematical Linguistics*, (64):5–24.
6. Hajičová, Eva, Petr Kuboň, and Vladislav Kuboň. 1990. Hierarchy of salience and discourse analysis and production. In *Proceedings of Coling'90*, Helsinki.
7. Jakubíček, Miloš, Aleš Horák, and Vojtěch Kovář. 2009. Mining phrases from syntactic analysis. In Václav Matoušek and Pavel Mautner, editors, *Text, Speech and Dialogue: 12th International Conference, TSD 2009, Pilsen, Czech Republic, September 13-17, 2009. Proceedings*, volume 5729 of *Lecture Notes in Computer Science*. Springer, Heidelberg, pages 124–130.
8. Kovář, Vojtěch, Aleš Horák, and Miloš Jakubíček. 2011. Syntactic analysis using finite patterns: A new parsing system for czech. In Zygmunt Vetulani, editor, *Human Language Technology. Challenges for Computer Science and Linguistics*, volume 6562 of *Lecture Notes in Computer Science*. Springer, Heidelberg, pages 161–171.
9. Lappin, Shalom and Herbert J. Leass. 1994. An algorithm for pronominal anaphora resolution. *Computatinal Linguistics*, 20(4):535–561.
10. Linh, Nguy Giang. 2006. Návrh souboru pravidel pro analýzu anafor v českém jazyce. Master's thesis, Charles University, Faculty of Mathematics and Physics, Prague.
11. Mitkov, Ruslan, Richard Evans, and Constantin Orăsan. 2002. A new, fully automatic version of Mitkov's knowledge-poor pronoun resolution method. In *Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2002)*, pages 168–186, Mexico City, Mexico, February, 17 – 23. Springer.
12. Müller, Christoph and Michael Strube. 2006. Multi-level annotation of linguistic data with MMAX2. In Sabine Braun, Kurt Kohn, and Joybrato Mukherjee, editors, *Corpus Technology and Language Pedagogy: New Resources, New Tools, New Methods*. Peter Lang, Frankfurt a.M., Germany, pages 197–214.
13. Novák, Michal and Zdeněk Žabokrtský. 2011. Resolving noun phrase coreference in czech. *Lecture Notes in Computer Science*, 7099:24–34.
14. Němčík, Vašek. 2011. Extracting Phrases from PDT 2.0. In Aleš Horák and Pavel Rychlý, editors, *Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2011*, pages 51–57, Brno. Tribun EU.
15. Šmerk, Pavel. 2007. Towards morphological disambiguation of czech. Ph.D. thesis proposal, Faculty of Informatics, Masaryk University.
16. Šmerk, Pavel. 2009. Fast morphological analysis of czech. In *Proceedings of the Raslan Workshop 2009*, pages 13–16, Brno. Masarykova univerzita.
17. Versley, Yannick, Simone P. Ponzetto, Massimo Poesio, Vladimir Eidelman, Alan Jern, Jason Smith, Xiaofeng Yang, and Alessandro Moschitti. 2008. BART: a modular toolkit for coreference resolution. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odjik, Stelios Piperidis, and Daniel Tapias, editors, *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, pages 962–965, Marrakech, Morocco. European Language Resources Association (ELRA).

Behaviour of the Czech Suffix *-ák* – A Case Study

Dana Hlaváčková, Karel Pala

NLP Centre
Faculty of Informatics, Masaryk University
Brno, Czech Republic
{hlavack,pala}@fi.muni.cz

Abstract. New techniques in Czech derivational morphology are discussed. They are based on the exploitation of the tool Deriv with integrated access to the main Czech dictionaries and corpora (SYN2000c and the new large Czech corpus CzTenTen12). The case study deals especially with the Czech suffix *-ák* – we describe its behaviour as completely as possible. The paper brings some new results in comparison with standard Czech grammars which do not rely on large data and software tools.

Key words: Czech morphology, Czech suffixes

1 Introduction

In the paper we report on an exploration of the derivational behaviour of the selected noun Czech suffixes *-ák, -ec, -ík,-ník*. We have used the tool Deriv (see `http://deb.fi.muni.cz/deriv`), which allows us to have a look at possibly all Czech nouns with the selected suffixes – here we will pay attention especially to the nouns with the suffix *-ák* that can be derived by the standard derivational processes. As far as we can say the previous exploration of the mentioned suffixes has been limited in number – because the authors of the standard Czech grammars ([1,2], further MČ and PMČ) did not have large corpora and machine dictionaries at their time, thus they were able to handle only a small number of examples.This inevitably meant that their results had to be only partial. The same can be said with regard to the Dokulil's [3] important work in which he laid the theoretical foundation of Czech word derivation but he did have sufficient data at his disposal.

2 Motivation

Our main purpose is to investigate the derivational relations in a more detailed and deeper way using larger and more complete Czech data. We are seeking to reach better coverage of the studied phenomena together with better precision. We start with the results that have been obtained for Czech formal morphology, namely with the morphological dictionary that is a part of Czech morphological analyzer Ajka [4]. Then we continue investigating derivational relations and

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2012, pp. 9–13, 2012.

behaviour of the selected suffixes in particular. As we use computer tools we need to make our description as formal as possible. We are convinced that better knowledge of derivational behaviour of the individual suffixes in Czech (and prefixes as well) is a necessary prerequisite for the development of the more inteligent searching tools that can be used in various applications, e.g. in searching engines for Web.

3 Computer processing of Czech word derivation

Obtaining better knowledge about the derivational relations in Czech requires larger data than was used so far in the mentioned standard Czech grammars [1,2]. Such data cannot be, however, reasonably processed manually, that would be too time consuming and would contain errors. At the moment we have at our disposal a large machine dictionary of Czech stems (approx. 400 000 items) whose coverage of Czech is about 95 %. To explore the behaviour of suffixes the tool Deriv has been developed, which makes it possible to formally describe the morphematic segmentation on the derivational level. This is handled with looking up the possible combinations of stems and suffixes (prefixes as well) also using regular expressions. In this way we obtain the required lists of nouns containing the particular suffixes as they occur in Czech. The tool Deriv [5] is integrated with two Czech corpora (SYN2000c [6] and CzTenTen12) as well as with the dictionary browser DEBDict, so the user (linguist) can see the behaviour of an explored suffix as completely as possible, namely its frequencies.

4 Starting data – morphological dictionary, corpora, ...

As we have said we work with the large machine dictionary of Czech stems (approx. 400 000 items) whose coverage of Czech is about 95 % (for comparison, the size of the SSJČ is twice smaller). It is a part of the Czech morphological analyzer Ajka [4] and the Deriv tool works with the lists of Czech stems generated appropriately for this purpose. The coverage 95 % means that the analyzer Majka is able to process any Czech text and recognize the word forms in it. Those 5 % are expressions consisting of numbers (dates, telephone numbers, etc.), e-mail addresses, URL's, words from other languages than Czech (most frequently Slovak and English) and others. Since the tool Deriv is linked with the two Czech corpora and six Czech dictionaries in Debdict the obtained results can be compared with them, especially with regard to the frequencies. The comparison of numbers obtained from the corpus SYN2000c with 114,363,813 and corpus CzTenTen12 with 5,414,437,666 tokens (see `http://ske.fi.muni.cz/auth/corpora/`) is then quite expressive. Thanks to links of the Deriv to the Word Sketch Engine also collocational data can be obtained. The data also contain the respective lists of Czech suffixes and prefixes.

5 Case study – behaviour of the Czech suffix *-ák*

The obtained list contains the derived nouns with the suffixem *-ák* which can be characterized in the following way:

- expressive and slang nouns, also obsolete ones
- nouns productive in deriving one word expressions

The number of the Czech nouns ending with the string *-ák* is 1351, from them there are 724 lemmata in masculine animate and 627 lemmata in masculine inanimate (they include also substandard lemmata not occurring in SSJČ and proper (family) names.

6 Derivational categories and their classification

For the nouns in the lists we propose the classification comprising the following categories with reference to the classifications that can be found in MČ and PMČ.

- Nouns derived from nouns:
  - agentive nouns - *dudák* (bagpiper), *koňák* (horseman), *sedlák* (farmer), *tramvaják* (tram driver)
  - nouns denoting inhabitants - *Brňák* (inhabitant of Brno), *Hanák* (inhabitant of Haná), *Malostraňák* (dweller of the Prague quarter Malá Strana), *Pražák* (inhabitant of Prague
  - nouns expressing membership in a group of people - *devětsilák* (member of the group Devětsil), *esenbák* (cop), *tatrovák* (worker in the Tatra factory)
  - nouns denoting animals (derived from feminines) - *lišák* (male fox), *myšák* (male mouse), *opičák* (male monkey)
  - augmentatives - *hnusák* (scumbag), *sviňák* (goat), *úchylák* (pervert)
- Nouns derived from adjectives:
  - nouns denoting bearers of properties - *blonďák* (blond man), *dobrák* (brick), *chudák* (poor fellow), *silák* (strongman)
- Nouns derived from numerals
  - nouns denoting order - *prvák* (first-year student), *druhák* (second-year student
  - nouns denoting roe deers and deers with regard to their antlers - *desaterák* (ten pointer), *dvanácterák* (royal stag)
- Nouns derived from verbs
  - agentive nouns - *divák* (viewer), *honák* (cowboy), *pašerák* (smuggler), *zpěvák* (singer
  - nouns denoting instruments - *bodák* (bayonet), *drapák* (grab), *hasák* (pipe-wrench), *naviják* (reel)

It is necessary to remark that number of the nouns in Table 1 is smaller than the numbers given above. This is due to the fact that we have put aside the nouns for which the reasonable derivational relation between the basic and derived form cannot be established or it is very difficult.

Table 1. The coverage of the nouns belonging to the individual categories

Category	Masculine animate	Masculine inanimate
agentive	98	-
inhabitants	78	-
groups	62	-
animals	16	-
augmentatives	47	-
bearers of property	128	
order	5	-
antlers	7	-
agentive1	105	-
instruments		-
Total	**546**	

7 Results and Conclusions

In the paper we paid attention to some selected Czech noun suffixes for which we describe their derivational behaviour. It has to be stressed that we have concentrated on just one suffix, namely *-ák*, which serves as a pattern that can be applied to other mentioned suffixes as well. The concrete results (numbers) are, however, brought only for the *-ák*. On the other hand, it is obvious that this kind of description can be applied to all mentioned suffix.

The main result is Table 1 which shows what meanings *-ák* have, the basic analysis standing behind Table 1 has been performed manually. The classification offered in Table 1 in fact removes ambiguity of *-ák*, which a human user resolves easily but computer applications cannot work without it.

We would like to stress that more can be said about the behaviour of *-ák* and suffixes of the same type – the paper is one of the first exercises in this respect. It also has to be noted that the use of the tool Deriv and large data made it possible to offer the results which display a reasonable coverage.

Acknowledgments

This work has been partly supported by the Ministry of Education of the Czech Republic project No. LM2010013 (Lindat–Clarin – Centre for Language Research Infrastructure in the Czech Republic), and by the Czech Science Foundation under the project P401/10/0792.

References

1. Komárek, M.: Mluvnice češtiny I (Grammar of Czech I). Academia, Praha (1986)
2. Karlík, P., Grepl, M., Nekula, M., Rusínová, Z.: Příruční mluvnice češtiny. Lidové noviny (1995)

3. Dokulil, M.: Tvoření slov v češtině I (Word Derivation in Czech I). Nakladatelství ČSAV, Praha (1962)
4. Šmerk, P.: K počítačové morfologické analỳze češtiny. (2010)
5. Šmerk, P.: Deriv. (2009) Web application interface (in Czech), accessible at: `http://deb.fi.muni.cz/deriv`.
6. ICNC: Czech National Corpus – SYN2000. Institute of the Czech National Corpus, Praha (2000) Accessible at: `http://www.korpus.cz`.

Reproducing Czech Syntactic Parsing Results Published in CoNLL Tasks

Lucia Kocincová

NLP Centre, Faculty of Informatics,
Masaryk University, Botanická 68a, 602 00 Brno, Czech Republic
lucyia@mail.muni.cz

Abstract. In this paper, I describe the approach on reproducing MST Parser and MaltParser results for Czech dependency parsing which were published in several tasks of Conference on Computational Natural Language Learning. Firstly, I briefly describe basic principles of parsers. Then, I include features and options that need to be optimized while using the parsers to get the desired results on testing files as well as evaluation methodology used in my research. I also shortly mention hardware requirements for those, who would like to train their own model for Czech language parsing. Finally, I include the practical application of trained models for our approach.

Key words: syntactic analysis, parsing, Czech, parser evaluation

1 Introduction

Dependency parsing universally is nowadays very evolving field in natural language processing, as dependency parsed data are further used for higher layer analysis of natural language. Also, development of tagged treebanks enabled to analyse languages another way – using data-driven parsing. That is why every improvement is proudly presented and various workshops with NLP tasks are founded. One of the best-known is Conference on Computational Natural Language Learning (CoNLL, now part of Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, shortly EMNLP-CoNLL), where shared tasks are hold to challenge the participants in natural language learning systems. Each team is given the same training data, so then the evaluation results can be better compared.

2 CoNNL shared task results for Czech language

Czech language dependency parsing was part of CoNLL shared task in 2006 [1], 2007 [2] and 2009 [3] and the aim of this paper is to describe the process of the approach to get the results shown in Table 1, although it may not be possible to achieve the exact accuracy, as it is covered later in the chapter.

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2012, pp. 15–22, 2012.

Table 1. Results presented in CoNNL shared tasks.

best accuracy in year	**Labeled Accuracy**	**Unlabeled Accuracy**
2006	80.18	87.30
2007	80.19	86.28
2009	80.38	(not tested)
MaltParser (2006)	78.42	84.80
MST Parser (2006)	80.18	87.30

2.1 Reasons why results may not be reproduced exactly

At the beginning of the research, I was determined that I should simulate all conditions that were depicted in conference papers but after further study of various materials, my determination changed. There were decisive reasons that cause the fact that my results may, or precisely, can not be the same as gained in CoNNL tasks:

- different scoring metrics were used (e.g. punctuation was not counting)
- versions of parsers were not stated
- different training and/or evaluation data (e.g. old version of corpora) may be used[1]
- different data format was used in 2009

In addition, I decided to use the latest versions of parsers because of the fact, that trained models from old versions are no longer runnable under the new ones. Therefore, old version models could be counterproductive if parsing data for later applications, such identified in section 7.

3 MaltParser

MaltParser is a complex system for statistical dependency parsing developed by Johan Hall, Jens Nilsson and Joakim Nivre at Växjö University and Uppsala University in Sweden. Using MaltParser, an induced model can be generated from corpus and then this model can be used to parse new data [4].

Latest version 1.7.2 was released on 25th September 2012 and is distributed under open source licence[2]. The system is being developed in Java from version 1.0.0.

MaltParser's specification can be divided into three parts[3]:

- **Parsing algorithm**: is defined by a *transition system* which derives dependency trees, together with an *oracle* that is used for reconstruction of each valid transition sequence for dependency structures. Together, there are seven deterministic parsing algorithms:

[1] due to licence conditions, they are not available online

[2] full specification can be found at `http://www.maltparser.org/license.html`

[3] complete list of features with their options can be found at `http://www.maltparser.org/userguide.html`

- **Nivre's algorithm**: is a linear-time algorithm for projective dependency structures that can be run in two modes, either arc-eager (nivreeager) or arc-standard (nivrestandard).
- **Stack algorithm**: are a group of three different algorithms: for projective (stackproj) and non-projective trees (stackeager, stacklazy)
- **Covington's algorithm**: is a quadratic-time algorithm for unrestricted dependency structures which modes can be restricted to projective structures (covproj) or non-projective (covnonproj) structures.

The last two parsing algorithms available in parser are:

- **Planar algorithm**: is another linear-time algorithm but for planar dependency structures (ones that do not contain any crossing links)
- **2-Planar algorithm**: also linear-time algorithm but can parse 2-planar dependency structures (ones whose links may be crossed following a specific rule)

After varying parser algorithms, especially first ones mentioned above, Lazy Stack algorithm carried out best accuracies with both types of learning packages.

– **Learning algorithm**: parser includes two machine learning packages – LIBSVM, a type of support vector machines with kernels, and LIBLINEAR, a type of various linear classifiers learner.
  I tried various experiments with both packages to search which one gives best results in meaning of time, memory use and accuracy. As can be seen in Table 3, LIBSVM attained best accuracy and learning time was nearly 73 hours.
– **Feature model**: is an external XML file that specifies features of partially built dependency structure together with main data structures in the parser configuration. Default model, which depends on combination of machine learning package and parsing algorithm, can be also used.
  I experimented with a few feature models, either build in or obtained by MaltOptimizer[4]. The best performance in my experiment was accomplished by one made with MaltOptimizer.

For achieving state-of-the-art results, optimization of MaltParser is necessary. It is a non-trivial task because not only machine learning algorithm needs to be correctly configured but also various parsers adjustments require setup - I used splitting functionality for speeding up training and parsing time in case of LIBSVM (split column was CPOSTAG, split structure was Stack[0] and split threshold 1000). In addition, each learning algorithm has options itself that can be enabled [7].

Hardware issues when using MaltParser mainly depend on which learning algorithm is chosen – both can give state-of-art accuracy but in case of LIBSVM, less memory is required, whereas LIBLINEAR is much more faster than LIBSVM (for training and also for parsing, ranging from 2–5 hours for LIBLINEAR and 8–72 hours for LIBSVM).

[4] MaltOptimizer is a free available tool for better and easier optimization of MaltParser, online at http://nil.fdi.ucm.es/maltoptimizer/

4 MSTParser

MSTParser is a non-projective dependency parser based on searching maximum spanning trees over directed graphs and is being developed by Jason Baldrige and Ryan McDonald.

Parser from latest versions supports CoNNL format but it distinguishes in some minor extent when comparing the input data format which accepts MaltParser. The system can parse data in its own MST format which is much more simpler than CoNNL:

w_1	w_2	w_3	...	w_n	– n words of a sentence
p_1	p_2	p_3	...	p_n	– POS tags for each word
l_1	l_2	l_3	...	l_n	– labels of the incoming edge to each word
d_1	d_2	d_3	...	d_n	– position of each words parent

Each sentence in the format is represented by first three or all four lines, where each data is tab spaced and whole sentences are space separated.

Optimization of MSTParser includes options such as number of iteration epochs for training, specifying type of structures for setting parsing algorithm (either projective or non-projective), denoting order of features and option if punctuation should be included in Hamming loss calculation. In newer versions, it is possible to add confidence scores per-edge that mark the parser's confidence in correctness of a particular edge.

While experimenting, I also tried out configuration stated in distribution of MSTParser, but I achieved best accuracy using value 3 for training-k[5] and using more memory than stated to avoid errors caused by running out of memory.

Latest version 0.5.0 was released on 23th January 2012 and the whole system, developed in Java, is distributed under *Apache License V2.0*[6].

Hardware issues with MSTParser are bound with sufficient memory while training a model, more specifically, the parser need to get a specification of how much heap space can be taken – I used about 15GB of memory while creating forest from training file with 1 503 739 tokens.

5 Data

5.1 Training and evaluation data

For training and evaluation purposes, The Prague Dependency Treebank 2.0 (PDT 2.0) was used. The newer version was chosen, because of the fact, that last two mentioned CoNNL tasks based the training on it. Moreover, updated version is free of various errors, such as spelling mistakes and faults on morfological and analytical layer [6].

[5] as stated in documentation, the k value is for non-projective structures only approximate

[6] definition available at `http://www.apache.org/licenses/LICENSE-2.0.html`

I used data anotated on analytical layer, which can be described with following numbers[7]:

	overall	train data	dtest data	etest data
sentences	87 913	68 495 (77.9 %)	9 270 (10.5 %)	10 148 (11.5 %)
tokens	1 503 739	1 171 191 (77.9 %)	158 962 (10.6 %)	173 586 (11.5 %)

For training numerous models, strictly only train part of the DPT 2.0 was used, so the rest of it – development test (dtest) and evaluation test (etest), was kept as unseen data. The data were manually disambiguated.

Before running the parsers, the format of the data has to be firstly changed to the CoNNL format in which a sentence consists of ten columns where each is tab separated (overview of the column and data meaning can be seen in Table 2) and individual sentences are separated by a blank line.

Table 2. CoNLL format preciselly described

Column #	Name	Definition
1	ID	Token counter (starting at 1 for each new sentence)
2	FORM	Word form or punctuation symbol
3	LEMMA	Lemma of word form or an underscore if not available
4	CPOSTAG	Coarse-grained part-of-speech tag
5	POSTAG	Fine-grained part-of-speech tag or identical to the coarse-grained part-of-speech tag if not available
6	FEATS	Unordered set of syntactic and/or morphological features separated by a vertical bar
7	HEAD	Head of the current token, which is either a value of ID or zero (0) – there may be multiple tokens with an ID of zero
8	DEPREL	Dependency relation to the HEAD – the dependency relation may be meaningfull or simply 'ROOT'
9	PHEAD	Projective head of current token, which is either a value of ID or zero (0)
10	PDEPREL	Dependency relation to the PHEAD

5.2 Evaluation metrics

For evaluation, a script was written following basic metrics that shows real accuracy:

$$UA = \frac{correct_{head}}{all_{head}}$$

[7] exhausting and precise description of PDT 2.0 can be found at http://ufal.mff.cuni.cz/pdt2.0/doc/pdt-guide/en/html/ch03.html

$$LA = \frac{correct_{head} \;\; AND \;\; correct_{label}}{all}$$

Where Unlabeled accuracy (*UA*) is the ratio between correctly determined head column ($correct_{head}$) and all heads (all_{head}) and Labeled accuracy (*LA*) is the result of correctly determined head column AND correctly determined label column ($correct_{label}$) at the same row over all rows in evaluation data.

6 Results

I can bring some relevant results that show the approach of obtaining the results of CoNNL shared tasks is completed. Table 3 presents selected various combinations of learning and parsing algorithms with results for each accuracy, labeled and unlabeled achieved on dtest. Parsing algorithms in italics cover non-projective dependency structures.
In my experiment, I managed to successfully reproduce MaltParser results – I achieved even higher score in both metrics, labeled and unlabeled accuracy. However, results with MSTParser were not accomplished, as shows Table 4.

Table 3. Accuracy achieved with MaltParser on dtest so far

learning alg.	**parsing alg.**	**Unlabeled accuracy**	**Labeled Accuracy**
LIBLINEAR	nivrestandard	70.62	64.73
	covproj	71.43	80.13
	stackproj	79.67	73.99
	covnonproj	80.58	74.95
	stackeager	82.54	77.14
	stacklazy	83.17	77.74
LIBSVM	nivreeager	83.21	78.42
	nivrestandard	81.51	76.37
	stackproj	83.00	77.47
	stacklazy	**85.02**	**80.05**

Table 4. Accuracy achieved with MSTParser on dtest

Unlabeled accuracy	**Labeled Accuracy**
77.73	69.19
83.01	75.34
83.04	**75.39**

7 Further experiments and practical applications

Further experiments will follow this attempt by turning the parsers for even better performance as recently, higher accuracy (about 2% higher in meaning of LA and 1% of meaning of UA) was published with MaltParser [8] and MST-Parser [9].
The aim of the approach was not only to get the results but it is far more practical. Systems with trained models that got the best accuracy will be used for parsing corpora that will be further utilized for application in SketchEngine[8] which is a corpus query system used by various people, including lexicographers, computer linguists and researchers.

Acknowledgments

This work has been partly supported by the Ministry of the Interior of CR within the project VF20102014003 and by the Czech Science Foundation under the project P401/10/0792.

References

1. Buchholz, S., Marsi E.:CoNLL-X Shared Task on Multilingual Dependency Parsing, In: Proceedings of the Tenth Conference on Computational Natural Language Learning, pp. 149–164 (2006), published: Stroudsburg, PA, USA, online at `http://dl.acm.org/citation.cfm?id=1596276.1596305`
2. Nivre, J., Hall, J., Kübler, S., McDonald, R., Nilsson, J., Riedel, S., Yuret, D.,: The CoNLL 2007 Shared Task on Dependency Parsing, In: Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, published: Association for Computational Linguistics, Prague, Czech Republic, pp. 915–932 (2007), online at `http://www.aclweb.org/anthology/D/D07/D07-1096`
3. Hajič, J., Ciaramita, M., Johansson, R., Kawahara, D., Martí, M. A., Marquez, L., Meyers, A., Nivre, J., Padó, S., Štepánek, J., Straňák, P., Surdeanu, M., Xue, N., Zhang, Y.: The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task, pp. 1-18 (2009)
4. Nivre, J., Hall, J., Nilsson, J.: MaltParser: A data-driven parser-generator for dependency parsing, In: Proceedings of LREC-2006, pp. 2216–2219 (2006)
5. McDonald, R., Pereira, F., Ribarov, K., Hajič, J.: Non-projective dependency parsing using spanning tree algorithms, In: Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, pp. 523–530 (2005)
6. Hajič, J.: Complex Corpus Annotation: The Prague Dependency Treebank, published: Jazykovedný ústav Ľ. Štúra, SAV, Bratislava, Slovakia, 2004
7. Nivre, J., Hall, J.: A Quick Guide to MaltParser Optimization, online at `http://maltparser.org/guides/opt/quick-opt.pdf`

[8] http://sketchengine.co.uk/

8. Nivre, J. : Non-Projective Dependency Parsing in Expected Linear Time. In: Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pp. 351-359. Association for Computational Linguistics, Suntec, Singapore. (2009)
9. Novák, V., Žabokrtský Z.: Feature Engineering in Maximum Spanning Tree Dependency Parser. In: Proceedings of the 10th International Conference on Text, Speech and Dialogue. Západočeská univerzita, Plzeň, Czechia. Springer-Verlag Berlin Heidelberg, LNCS 4629. (2007)

Adaptation of Czech Parsers for Slovak

Marek Medveď, Miloš Jakubíček, Vojtěch Kovář, Václav Němčík

Natural Language Processing Centre
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
{xmedved1,jak,xkovar3,xnemcik}@fi.muni.cz

Abstract. In this paper we present an adaptation of two Czech syntactic analyzers `Synt` and `SET` for Slovak language. We describe the transformation of Slovak morphological tagset used by the Slovak development corpora `skTenTen` and `r-mak-3.0` to its Czech equivalent expected by the parsers and modifications of both parsers that have been performed partially in the lexical analysis and mainly in the formal grammars used in both systems. Finally we provide an evaluation of parsing results on two datasets – a phrasal and dependency treebank of Slovak.

Key words: syntactic analysis, parsing, Slovak

1 Introduction

Czech and Slovak are both representatives of Slavonic free-word-order languages with rich morphology. The most differences between Czech and Slovak lie in the lexicon – on morphological and even more on syntactic level both languages are very similar. Currently, there is no full parser available for Slovak, only tools that produce partial analysis based either on regular expressions [1] or predicate-argument structure [2]. Because of the syntactic similarity of these languages, we took the opportunity to adjust two currently available Czech parsers, `Synt` and `SET`, for Slovak.

Syntactic analyzers `Synt`[3] and `SET`[4] have been developed over the past years in the Natural Language Processing Centre at Faculty of Informatics, Masaryk University. Both systems are rule-based but take a different approach to the challenges of syntactic analysis. The `Synt` parser is based on a context-free backbone enhanced with contextual actions and performs a stochastic agenda-based head-driven chart analysis. The syntactic parser `SET` is based on a simple grammar consisting of regular expressions over morphological tags and performs segmentation of input sentence according to the grammar rules.

The input for both `Synt` and `SET` is a sentence in the form of vertical text morphologically annotated by the morphological analyzer `Ajka`[5] which uses an attributive tagset described in [6][1].

[1] Current version of the tagset is available online at `http:///nlp.fi.muni.cz/ma/`.

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2012, pp. 23–30, 2012.

The output of Synt may be:

- **a phrase-structure tree**
  This is the main output of Synt and consists of a set of phrase-structure trees ordered according to the tree ranking. The system makes it possible to retrieve n-best trees effectively.
- **a dependency graph**
  A dependency graph represents a packed structure which can be utilized to extract all possible dependency trees. It is created by using the head and dependency markers that might be tied with each rule in the grammar.
- **set of syntactic structures**
  The input sentence is decomposed into a set of unambiguous syntactic structures chosen by the user.

The output of SET may be:

- **a hybrid tree** consisting of both dependency and constituent edges,
- **a pure dependency tree**,
- **a pure constituent tree**.

In our experiments, we used three Slovak corpora as input for the parsers – the r-mak 3.0[**?**] corpus, containing 1.2M tokens and manual morphological annotation and the skTenTen corpus[7], a large web corpus containing about 876M tokens with automatic morphological annotation, and a subset of a Slovak dependency treebank[8] that is currently under development in the Slovak Academy of Sciences, which contained more than 12,000 sentences and is further referred to as SDT.

For the parsers to be able to process the Slovak input, the following modifications had to be performed:

- **morphological tagging conversion** into the format expected by the parsers,
- **lexical analysis adjustment** in both parsers (e.g. mapping of lexical units to grammar non-terminals),
- **grammar adaptation** for both parsers, covering syntactic phenomena in which Czech and Slovak are different.

2 Morphological tagging conversion

In this section we describe the translation from the Slovak tagset to its counterpart in the Czech tagset and explain the steps necessary for correct function of syntactic analyzers. Both r-mak 3.0 and skTenTen use a positional tagset.[2] For the purpose of converting the annotation into the format given by the Czech morphological analyser Ajka, a translation script has been created called sk2cs.py, which takes a vertical text as input and translates each tag to its Czech equivalent.

[2] Available online at http://korpus.sk/morpho.html.

Obviously, there is no 1:1 mapping between tags in the tagsets, e.g. due to different subclassification paradigms for several part-of-speech (PoS) kinds. Therefore the translation process consists of three steps:

1. **rule-based translation**
2. **whitelist translation**
3. **whitelist completion**

2.1 Ruled-based tag translation

At first, the input tag is translated using a predefined set of rules that map each grammatical category to its counterpart in the Czech tagset. If this mapping is ambiguous (1:n), the program either just chooses the first tag or, optionally, produces an ambiguous output.

2.2 Whitelist-based tag translation

For words where the PoS of the Slovak tag is different than the one of its Czech equivalent, a whitelist-based procedure is used that directly maps selected words to their Czech tags. An example of a problematic translation is the word *mnoho* (a lot of) which is said to be an adverb in Czech but a numeral in Slovak. It should be noted that the morphological classification of this word (and many others) is a cumbersome issue with no clear solution, and we do not claim that the Czech or Slovak classification is better than the other one.

2.3 Whitelist-based tag completion

Finally, in some cases the Slovak tagset is less fine-grained than the Czech one and the resulting tag would not contain enough information for the parsing to be successful. This concerns e.g. pronouns for which the Slovak tagset does not contain any subclassification that would distinguish relative, interrogative and demonstrative pronouns, but both parsers use this kind of information in their grammar. Fortunately, the sets of all these pronouns are small enough to be handled case-by-case as well, and therefore the translation process uses another whitelist to extend the morphological annotation for them.

3 Adaptation of `Synt`

`Synt` is a rule-based parser consisting of a context-free grammar (CFG) enhanced by in-programmed contextual actions for capturing contextual phenomena like e.g. grammatical agreement. The parsing process consists of two steps: first a basic chart parsing is performed using the CFG and producing a large set of candidate analyses in the form of the resulting chart – a packed forest of trees. On top of the chart, the contextual actions are evaluated, pruning the analyses space by orders of magnitude and producing final parsing results.

To prevent maintenance issues a rule-based system may suffer from, the grammar is developed in the form of a meta-grammar, consisting of only about 250 rules. From this meta-grammar a full grammar is automatically derived by exploiting per-rule defined derivation actions (e.g. expanding a group of non-terminals or permutating right-hand side of a meta-rule).

The modifications for Slovak in `Synt` consist from two parts:

- **adaptation of lexical analysis**
- **adaptation of grammar rules**

3.1 Lexical analysis adaptation

In `Synt` lexical analysis is a process that assigns a pre-terminal (i.e. last non-terminal in the tree that is directly rewritten to the surface word) to a given word by using word's morphological classification. In some cases (e.g. identification of some named-entities like months, or specific handling of modal verbs), the lexical analysis exploits not only the tag of the word, but also its lemma or the word itself. In these cases the analysis had to be modified (translated) to Slovak.

3.2 Grammar rules adaptation

In the following we list a number of syntactic phenomena that need to be handled differently in Czech and Slovak.

Sentences with passive Expression of passive in Slovak is different from Czech. The Czech passive structure is: ***to be + passive verb*** (figure 1). But in Slovak the structure is: ***to be + adjective***.

```
clause %> is vpasr
vpasr -> VPAS
```

Fig. 1. Original rule for passive.

Therefore it is necessary to adapt this rule (figure 2). The adaptation consists of replacing pre-terminal `VPAS` by pre-terminal `ADJ`.

```
clause %> is vpasr
vpasr -> ADJ
```

Fig. 2. Adapted rule for passive in Slovak language.

Sentences with structure *not + to be* This structure shows the main difference between Slovak and Czech. In Slovak (figure not2) this structure is expressed by two words but in Czech language it is expressed only by one word.

```
Original:          clause %> IS sth
                   clause %> ARE sth
                   clause %> VB12 sth
                   -------------------
Adapted:           clause %> is sth
                   clause %> are sth
                   clause %> vb12 sth
                   is -> IS
                   is -> IS NOT
                   are -> ARE
                   are -> ARE NOT
                   vb12 -> VB12
                   vb12 -> NOT VB12
```

Fig. 3. Adaptation of Czech rule for Slovak structure ***not + to be***

Sentences with structure *would + to be* The same case as structure ***not + to be*** is the structure ***would + to be***. The modification of this rule (figure4) divides one word into two words with same semantics.

```
Original:          clause %> VBK sth
                   -----------------
Adapted:           clause %> vbk sth
                   vbk -> VBK
                   vbk -> VBK VB12
```

Fig. 4. Structure ***would + to be***

Sentences with structure *if + to be* or *that + to be*
The next case of Slovak structure which contains two divided words instead of one word expression is structure ***if + to be*** or ***that + to be***. The new rule describing this two structures is on figure 5.

Sentences with multiple numbers For sentences with structure „three times“ there was no pre-terminal for word „times“ which is written separately in Slovak. A new rule associated with this pre-terminal was created too. This new rule can analyzed structure ***„three times“***, or structure ***„3 times“***(figure 6).

```
Original:          clause %> akvbk sth
                   akvbk -> KVBK
                   akvbk -> AVBK
                   -------------------
Adapted:           clause %> akvbk sth
                   akvbk -> KVBK is
                   akvbk -> AVBK is
                   akvbk -> KVBK are
                   akvbk -> AVBK are
                   akvbk -> KVBK vb12
                   akvbk -> AVBK vb12
```

Fig. 5. Rule for structure ***if + to be*** and ***that + to be*** (sk)

```
numk -> NUMK TIMES
```

Fig. 6. Added pre-terminal `TIMES`

3.3 Adaptation of `SET`

`SET` is based on a simple grammar mostly consisting of regular expressions over morphological tags. Similarly to `Synt`, the grammar is directly lexicalized in some cases and required appropriate modifications. Besides the lexical analysis, following changes have been performed to the grammar:

Structure *would + to be*
In the same way as in `Synt`, the Czech expression for this structure had to be divided into two words (figure 7).

```
TMPL: $PARTICIP $...* $BY $BYBYT MARK 2 DEP 0    PROB 1000
%$BYBYT(word): som si sme ste
%TMPL: $BY $BYBYT MARK 1 DEP 0 PROB 1000
```

Fig. 7. Structure ***would + to be***

Structure *not + to be*
The same situation as before is in this case (figure 8).

4 Evaluation

The modifications have been evaluated for both parsers separately. For `Synt`, the coverage was measured on two corpora, the `r-mak 3.0` and `SDT`. To convert

```
%TMPL: $PARTICIP $...* $NOT $BYBYT MARK 2 DEP 0    PROB 1000
%$BYBYT(word): som si sme ste
```

Fig. 8. Structure ***not + to be***

the SDT treebank from its native XML format into annotated vertical text, the `pdt2vert`[9] was used. The precision of `Synt` was measured on a random sample of 77 sentences from the `skTenTen` corpus that were accepted by the parser and for which a correct constituent tree was determined. The LAA tree similarity metric [10] was used for the evaluation.

Since `SET` always produces some dependency tree, only dependency precision was evaluated against the SDT.

4.1 Evaluation of `Synt` parser

Corpus	Number of sentences	Number of accepted
r-mak 3.0	74,127	77 %
SDT	12,762	76.9 %

Table 1. Evaluation of the coverage of `Synt`

Number of sentences	77
Median number of trees	148
Average number of trees	71595.81
Average LAA of the first tree	87.13
Time per sentence	0.038 s

Table 2. Evaluation of the precision of `Synt`

4.2 Evaluation of `SET` parser

Corpus	Number of sentences	Dependency precision
SDT	12,762	56.7 %

Table 3. Evaluation of the precision of `SET`

5 Conclusions and Future Development

In this paper we have presented two Czech parsers, `Synt` and `SET`, adapted for Slovak. These represent first full parsing solutions available for Slovak. In the future further development of both parsers on Slovak is planned towards better precision and coverage on larger datasets.

Acknowledgements

We hereby thank Radovan Garabík, Ľudovít Štúr Institute of Linguistics, Slovak Academy of Sciences for his kind help and willingness to provide us with development and evaluation data. The work has been partly supported by the Ministry of Education of CR within the LINDAT-Clarin project LM2010013.

References

1. Trabalka Marek, B.M.: Realization of syntactic parser for inflectional language using XML and regular expressions. In: Text, Speech and Dialogue, volume 1902 of Lecture Notes in Computer Science, (Springer Berlin / Heidelberg) pages 59–90
2. Ondáš Stanislav, J.J., Čižmár Anton: Extracting sentence elements for the natural language understanding based on Slovak national corpus. (In: Analysis of Verbal and Nonverbal Communication and Enactment. The Processing Issues, volume 6800 of Lecture Notes in Computer Science)
3. Jakubíček, M., Horák, A., Kovář, V.: Mining phrases from syntactic analysis. In: Text, Speech and Dialogue. (2009) 124–130
4. Kovář, V., Horák, A., Jakubíček, M.: Syntactic analysis using finite patterns: A new parsing system for czech. In: Human Language Technology. Challenges for Computer Science and Linguistics, Berlin/Heidelberg (2011) 161–171
5. Šmerk, P.: Fast Morphological Analysis of Czech. In: Proceedings of the Raslan Workshop 2009, Brno (2009)
6. Jakubíček, M., Kovář, V., Šmerk, P.: Czech Morphological Tagset Revisited. Proceedings of Recent Advances in Slavonic Natural Language Processing 2011 (2011) 29–42
7. Masaryk University, Lexical Computing Ltd.: skTenTen – Slovak web corpus (2011) `http://trac.sketchengine.co.uk/wiki/Corpora/skTenTen`, [Online].
8. Gajdošová, K.: Syntaktická anotácia vybraných textov slovenského národného korpusu. In Múcsková, G., ed.: Varia. 16. Zborník materiálov zo XVI. kolokvia mladých jazykovedcov, Slovak Linguistic Society, Ľudovít Štúr Institute of Linguistics, Slovak Academy of Sciences (2009) s. 140 – 148
9. Němčík, V.: Extracting Phrases from PDT 2.0. In Horák, A., Rychlý, P., eds.: Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2011, Brno, Tribun EU (2011) 51–57
10. Sampson, G., Babarczy, A.: A test of the leaf-ancestor metric for parse accuracy. Natural Language Engineering **9**(04) (2003) 365–380

Part II

Logic and Language

Deduction System for TIL-2010

Marie Duží[1], Marek Menšík[1,2], Lukáš Vích[1]

[1] VŠB - Technical university Ostrava
17. listopadu 15, 708 33 Ostrava, Czech republic
[2] Silesian university in Opava,
Bezrucovo namesti 13, 746 01 Opava, Czech Republic

Abstract. The goal of this paper is to introduce a deductive system for Tichý's Transparent Intensional Logic (TIL). Tichý defined a sequent calculus for pre-1988 TIL, that is TIL based on the simple theory of types. Thus we first briefly recapitulate the rules of this simple-type calculus. Then we describe the adjustments of the calculus so 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. We show that operating in a hyperintensional context is far from being technically trivial. Yet it is feasible. To this end we introduce a substitution method that operates on hyperintensions. The syntax of TIL is the typed lambda calculus. Its semantics is based on a procedural redefinition of, inter alia, functional abstraction and application. The only two non-standard features are a hyperintension (called *Trivialization*) that presents objects, including hyperintensions, and a four-place substitution function (called *Sub*) defined over hyperintensions.

Key words: Transparent Intensional Logic, TIL, deductive system, inference

1 Foundations of TIL

From the formal point of view, TIL is a hyperintensional, partial typed λ-calculus. Thus the syntax of TIL is Church's (higher-order) typed λ-calculus, but with the all-important difference that the syntax has been assigned a procedural (as opposed to denotational) semantics, according to which a linguistic sense is an abstract *procedure* detailing how to arrive at an object of a particular logical type. TIL *constructions* are such procedures. Thus, abstraction transforms into the molecular procedure of forming a function, application into the molecular procedure of applying a function to an argument, and variables into atomic procedures for arriving at their assigned values.

There are two kinds of constructions, atomic and compound (molecular). Atomic constructions (*Variables* and *Trivializations*) do not contain any other constituent but themselves; they specify objects (of any type) on which compound constructions operate. The *variables* x, y, p, q, ..., construct objects dependently on a valuation; they v-construct. The *Trivialisation* of an object X (of

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2012, pp. 33–41, 2012.

any type, even a construction), in symbols $^{0}\boldsymbol{X}$, constructs simply X without the mediation of any other construction. *Compound* constructions, which consist of other constituents as well, are *Composition* and *Closure*. *Composition* $[\boldsymbol{F}\ \boldsymbol{A_1}\ldots\boldsymbol{A_n}]$ is the operation of functional application. It v-constructs the value of the function f (*valuation-*, or *v-*, –constructed by F) at a tuple – argument A (v-constructed by $A_1, \ldots, A_n$), if the function f is defined at A, otherwise the Composition is *v-improper*, i.e., it *fails* to v-construct anything.[3] *Closure* $[\lambda\boldsymbol{x_1}\ldots\boldsymbol{x_n}\ \boldsymbol{X}]$ spells out the instruction to v-construct a function by abstracting over the values of the variables $x_1,\ldots,x_n$ in the ordinary manner of the λ-calculi. Finally, higher-order constructions can be used twice over as constituents of composite constructions. This is achieved by a fifth construction called *Double Execution*, $^{2}\boldsymbol{X}$, that behaves as follows: If X v-constructs a construction X', and X' v-constructs an entity Y, then ^{2}X v-constructs Y; otherwise ^{2}X is *v-improper*, failing as it does to v-construct anything.

TIL constructions, as well as the entities they construct, all receive a type. The formal ontology of TIL is bi-dimensional; one dimension is made up of constructions, the other dimension encompasses non-constructions. On the ground level of the type hierarchy, there are non-constructional entities unstructured from the algorithmic point of view belonging to a *type of order 1*. Given a so-called *epistemic (or objectual) base* of *atomic types* (o-truth values, ι-individuals, τ-time moments / real numbers, ω-possible worlds), the induction rule for forming functional types is applied: where α, $\beta_1,\ldots,\beta_n$ are types of order 1, the set of partial mappings from $\beta_1 \times \ldots \times \beta_n$ to α, denoted '$(\alpha\beta_1\ldots\beta_n)$', is a type of order 1 as well.[4] Constructions that construct entities of order 1 are *constructions of order 1*. They belong to a *type of order 2*, denoted '$*_1$'. The type $*_1$ together with atomic types of order 1 serves as a base for the induction rule: any collection of partial mappings, type $(\alpha\beta_1\ldots\beta_n)$, *involving* $*_1$ in their domain or range is a *type of order 2*. Constructions belonging to a type $*_2$ that identify entities of order 1 or 2, and partial mappings involving such constructions, belong to a *type of order 3*. And so on *ad infinitum.*

The principle of hyperintensional individuation would slot in between Church's Alternatives (0) and (1) as Alternative (3/4), in that α-conversion and η-conversion together with a restricted principle of β-conversion determine the procedural individuation of hyperintensions we are operating with.

Laying out the required semantics requires a fair amount of footwork. Once this is in place, however, all that remains is filling in the nitty-gritty details of extensional rules such as quantifying-into hyperintensional contexts and substitution of identicals. The devil is in the detail, as ever, and defining

[3] We treat functions as partial mappings, i.e., set-theoretical objects, unlike the *constructions* of functions.

[4] TIL is an open-ended system. The above epistemic base {o, ι, τ, ω}was chosen, because it is apt for natural-language analysis, but the choice of base depends on the area and language to be analysed. For instance, possible worlds and times are out of place in case of mathematics, and the base might consist of, e.g., o and ν, where ν is the type of natural numbers.

extensional rules of inference for hyperintensional contexts is far from being technically trivial. But it is feasible, which we are going to show in the rest of the paper. When defining extensional rules for operating in (hyper-)intensional contexts we encounter two main problems, namely the problem of *substitution* of identicals (Leibniz) and *existential generalization*. A common idea is that extensional (etc.) contexts are those that validate quantifying-in and substation of identicals. And conversely, if a context resists some of these rules, it is deemed to be in violation of one or more of the laws of extensional logic and as eluding full logical analysis. What we are saying is that also intensional and hyperintensional contexts may be quantified into, but that the feasibility of doing so presupposes that it be done within an extensional logic of hyperintensional contexts.

2 Tichý's sequent calculus

Tichý proposed a solution of the substitution and existential generalization problem in his (1982, 1986) and defined a sequent calculus for the pre-1988 TIL, that is for extensional and intensional contexts. The solution is restricted to the so-called linguistic constructions of the form $\lambda w \lambda t[C_1 C_2 \ldots C_m]$ or $\lambda w \lambda t[\lambda x_1 \ldots x_m C]$.

2.1 Substitution and existential generalization

a) *Substitution*. $a = b; C(a/x) \vdash C(b/x)$
This rule seems to be invalid in intensional contexts. For instance, the following argument is obviously invalid:

The President of ČR is the husband of Livie.
Miloš Zeman wants to be the President of ČR.

Miloš Zeman wants to be the husband of Livie.

b) *Existential generalization*. $C(a/x) \vdash \exists x C(x)$
Again, in intensional contexts this rule seems to be invalid. For instance, the following argument is obviously invalid:

Miloš Zeman wants to be the President of ČR.

The President of ČR exists.

Ad a) Tichý defines in (1986) *hospitality* for substitution. In principle, there are four cases. If a variable z is (1,1) hospitable, then the construction of the form $[X_{wt}]$ is substitutable for z. That is, z occurs in an extensional (*de re*) context. If a variable z is (1,0) hospitable, then the construction of the form $[X\ \boldsymbol{w}]$ is substitutable for z. That is, z occurs in an intensional (*de dicto*) context with respect to time t. If a variable z is (0,1) hospitable, then the construction of the form $[X\ \boldsymbol{t}]$ is substitutable for z. That is, z occurs in an intensional (*de dicto*) context with respect to a world w. Finally, if a variable z is (0,0) hospitable, then the construction of the form X is substitutable for z. That is, z occurs in an intensional (*de dicto*) context with respect to both t and w.

Ad b) Exposure and *existential generalization*. Let x be (1,1)-hospitable, $D(k,l)$ substitutable for x in C. Then the following rule is valid:

$$C(D(k,l)/x) \vdash \lambda w\lambda t\exists x\, C(x)$$

Example. $\lambda w\lambda t\ [Ekonom_{wt}\ PCR_{wt}] \vdash \lambda w\lambda t\exists x[Ekonom_{wt}\, x]$; ($Ekonom/(o\iota)_{\tau\omega}$; $PCR/\iota_{\tau\omega}$; $x \to_v \iota$.)

2.2 Sequent calculus

Basic notions we need are these.

Match is a pair $a : C$, where $a, C \to \alpha$ and a is an atomic construction. A match a:C is *satisfied* by a valuation v, if a and C v-construct the same object; match :C is satisfied by v, if C is v-improper; matches a:C # b:C are *incompatible,* if a, b construct different objects; matches a:C # :C are *incompatible.*

Sequent is a tuple of the form a_1:C_1, ..., a_m:$C_m \to b$:D, for which we use a generic notation $\Phi \to \Psi$; A sequent $\Phi \to \Psi$ is *valid* if each valuation satisfying Φ satisfies also Ψ;

Next we define rules preserving validity of sequents.

Structural rules.

1. $\| \Phi \to \Psi$, if $\Psi \in \Phi$ (trivial sequent)
2. $\Phi \to \Psi \| \Phi_s \to \Psi$, if $\Phi \subseteq \Phi_s$ (redundant match)
3. $\Phi, \vartheta \to \Psi; \Phi \to \vartheta \| \Phi \to \Psi$ (simplification)
4. $\| \Phi \to$ y:y (trivial match)
5. $\Phi \to \vartheta 1; \Phi \to \vartheta 2 \| \Phi \to \Psi$, if $\vartheta 1$ and $\vartheta 2$ are incompatible
6. $\Phi, :\vartheta \to \Psi; \Phi, y{:}\vartheta \to \Psi \| \Phi \to \Psi$($y$ is not free in ...)

Application rules.

7. *a-instance* (modus ponens):

$\Phi \to y{:}[FX_1 \ldots X_m]$, Φ, $f{:}F$, $x_1{:}X_1,\ldots,x_m{:}X_m \to \Psi \| \Phi \to \Psi$, ($f$, x_i, different variables, free in Φ, Ψ, F, X_i)

8. *a-substitution*:
   (i) $\Phi \to y{:}[FX_1 \ldots X_m], \Phi \to x_1{:}X_1,\ldots,\Phi \to x_m{:}X_m \| \Phi \to y{:}[Fx_1 \ldots x_m]$
   (ii) $\Phi \to y{:}[Fx_1 \ldots x_m]; \Phi \to x_1{:}X_1,\ldots, \Phi \to x_m{:}X_m \| \Phi \to y{:}[FX_1 \ldots X_m]$
9. *extensionality*:

$\Phi, y{:}[f x_1 \ldots x_m] \to y{:}[g x_1 \ldots x_m]; \Phi, y{:}[g x_1 \ldots x_m] \to y{:}[f x_1 \ldots x_m] \| \Phi \to f{:}g$
($y, x_1,\ldots,x_m$ are different variables that are not free in Φ, f, g.)

λ-rules.

10. $\Phi, f{:}\lambda x_1 \ldots x_m Y \to \Psi \| \Phi \to \Psi$($f$ is not free in Φ, Y,Ψ)
11. β-reduction:
    $\Phi \to y{:}[[\lambda x_1 \ldots x_m Y]\ X_1 \ldots X_m] \|$
    $\Phi \to y{:}Y(X_1 \ldots X_m/x_1 \ldots x_m)$; ($X_i$ is substitutable for x_i)
12. β-expansion:
    $\Phi \to x_1{:}X_1;\ldots;\Phi \to x_m{:}X_m; \Phi \to y{:}Y(X_1 \ldots X_m/x_1 \ldots x_m) \|$
    $\Phi \to y{:}[[\lambda x_1 \ldots x_m Y]\ X_1 \ldots X_m]$

3 Generalization for TIL 2010

Our goal is to generalize the calculus so that it involves *ramified theory of types*, all kinds of constructions, existential generalization to any contexts and substitution of identicals in any kind of context. To this end we first specify the free kinds of context.[5]

3.1 Three kinds of context

Constructions are full-fledged objects that can be not only used to construct an object (if any) but also serve themselves as input/output objects on which other constructions (of a higher-order) operate. Thus we have:

Hyperintensional context: the sort of context in which a construction is not used to v-construct an object. Instead, the construction itself is an argument of another function; the construction is just mentioned.

Example. "Charles is solving the equation $1 + x = 3$". When solving the equation, Charles wants to find out which set (here a singleton) is constructed by the Closure $\lambda x[^0= [^0+ \ ^0 1 \ x] \ ^0 3]$. Hence this Closure must occur hyperintensionally, becuase Charles is related to the Closure itself rather than its product, a particular set. Otherwise the seeker would be immediately a finder and Charle's solving would be a pointless activity. The analysis comes down to:

$$\lambda w \lambda t[^0 Solve_{wt} \ ^0 Charles \ ^0[\lambda x[^0= [^0+ \ ^0 1 \ x] \ ^0 3]]].$$

Intensional context: the sort of context in which a construction C is used to v-construct a function f but not a particular value of f; moreover, C does not occur within another hyperintensional context.

Example. "Charles wants to be The President of Finland". Charles is related to the office itself rather than to its occupier, if any. Thus the Closure $\lambda w \lambda t[^0 President\_of_{wt} \ ^0 Finland]$ must occur intensionally, because it is not used to v-construct the holder of the office (particular individual, if any). The sentence is assigned as its analysis the construction

$$\lambda w \lambda t[^0 Want\_to\_be_{wt} \ ^0 Charles \ \lambda w \lambda t[^0 President\_of_{wt} \ ^0 Finland]].$$

Extensional context: the sort of context in which a construction C of a function f is used to construct a particular value of f at a given argument, and C does not occur within another intensional or hyperintensional context.

Example. "The President of Finland is watching TV". The analysis of this sentence comes down to the Closure

$$\lambda w \lambda t[^0 Watch_{wt} \ \lambda w \lambda t[^0 President\_of_{wt} \ ^0 Finland]_{wt} \ ^0 TV].$$

The meaning of 'the President of Finland' occurs here with *de re* supposition, i.e. extensionally.

[5] For exact definitions see [5, §2.6] and also [2, Chapter 11].

3.2 Extensional calculus of hyperintensions

First we specify the rules of existential generalization and substitution rules for all kinds of context and for any constructions. In order to operate in hyperintensional context we need to introduce a four-place substitution function, $Sub/(*_n\, *_n\, *_n *_n)$, defined over hyperintensions. When applid to constructions C_1, C_2 and C_3 the function returns as its value the construction D that is the result of correctly substituting C_1 for C_2 into C_3.

Let $F/(\alpha\beta)$; a/α. First we specify the ***rules for existential generalisation***.[6]

a) extensional context.

Let an occurrence of the Composition $[\dots[^0F\ ^0a]\dots]$ be extensional and let it v-construct the truth-value **T**. Then the following rule is valid:

$$[\dots[^0F\ ^0a]\dots] \vdash \exists x[\dots[^0Fx]\dots]; x \to_v \alpha$$

Example. „Pope is wise." $\models$ „Somebody is wise".

$$\lambda w\lambda t[^0Wise_{wt}\ ^0Pope_{wt}] \models \lambda w\lambda t\exists x[^0Wise_{wt}\ x];$$

b) intensional context.

Let $[^0F\ ^0a]$ occur intensionally in $[\dots\lambda y\ [\dots[^0F\ ^0a]\dots]]$ that v-constructs **T**. Then the following rule is valid:

$$[\dots\lambda y[\dots[^0F\ ^0a]\dots]] \vdash \exists f[\dots\lambda y[\dots[f\ ^0a]\dots]]; f \to_v (\alpha\beta)$$

Example. „b believes that Pope is wise". $\models$ „There is an office such that b believes that its holder is wise".

$$\lambda w\lambda t[^0Believe_{wt}\ ^0b\ \lambda w\lambda t[^0Wise_{wt}\ ^0Pope_{wt}]] \models$$
$$\lambda w\lambda t\exists f[^0Believe_{wt}\ ^0b\ \lambda w\lambda t[^0Wise_{wt}\ f_{wt}]];$$

c) *hyperintensional* context.

Let $[^0F\ ^0a]$ occur hyperintensionally in a construction $[\dots{}^0[\dots[^0F\ ^0a]\dots]]$ that v-constructs **T**. Then the following rule is truth-preserving:

$$[\dots{}^0[\dots[^0F\ ^0a]\dots]] \vdash \exists c\ ^2[^0Sub\ c\ ^{00}F\ ^0[\dots{}^0[\dots[^0F\ ^0a]\dots]]];$$
$$c \to_v *_n;\ ^2c \to_v (\alpha\beta)$$

Example. „b believes* that Pope is wise." $\models$ „There is a concept of an office such that b believes* that the holder of the office is wise."

$$\lambda w\lambda t[^0Believe^*_{wt}\ ^0b\ ^0[\lambda w\lambda t[^0Wise_{wt}\ ^0Pope_{wt}]] \models$$
$$\lambda w\lambda t\exists c[^0Believe^*_{wt}\ ^0b\ [^0Sub\ c\ ^{00}Pope\ ^0[\lambda w\lambda t[^0Wise_{wt}\ ^0Pope_{wt}]]]];$$
$$(Believe^*/(o\iota *_n)_{\tau\omega} : \text{hyperpropositional attitude}; c \to_v *_n;\ ^2c \to_v \iota_{\tau\omega}.)$$

[6] For details see [1].

Second, here are the ***rules for substitution (Leibniz).***

a) In an *extensional context* substitution of ***v-congruent*** constructions is valid.
b) In an *intensional context* (modalities, notional attitudes, ...) substitution of ***equivalent*** (but not only v-congruent) constructions is valid.
c) In a *hyperintensional context* (propositional attitudes, mathematical sentences, ...) substitution of ***procedurally isomorphic*** (but not only equivalent) constructions is valid.

Third, we must specify how to manage partial functions, that is ***compositionality*** and ***non-existence.*** If a function F has no-value at an argument a (value gap) then the Composition $[^0F\ ^0a]$ is *v-improper*, and so is any construction C occurring extensionally and containing $[^0F\ ^0a]$ as a constituent; *partiality is strictly propagated up*:

$[\ldots[\ldots[^0F\ ^0a]\ldots]\ldots]$ is *v-improper* until the context is raised up to *hyper/intensional* level:

intensional context : $\lambda x\ldots[\ldots[\ldots[^0F\ ^0a]\ldots]\ldots]$ is *v-proper*
hyperintensional context: $^0[\ldots[\ldots[^0F\ ^0a]\ldots]\ldots]$ is *v-proper*

The ***rules of sequent calculus*** remain as specified by Tichý with one important exception. Tichý's λ-rules involve β-reduction 'by name'. This rule is validity preserving, but we need a stronger rule that would guarantee equivalency between redex and reduct in the sense that both either v-construct the same object or both are v-improper. Moreover, β-reduction 'by name' can yield a loss of analytic information.[7]

β-reduction 'by name' in the sequent calculus:

$\Phi \rightarrow y{:}[[\lambda x_1\ldots x_m Y]\ X_1\ldots X_m] \parallel \Phi \rightarrow y{:}Y(X_1\ldots X_m/x_1\ldots x_m)$; ($X_i$ is substitutable for x_i)

In logic of *partial functions* the rule of transformation $[[\lambda x_1\ldots x_m Y]\ X_1\ldots X_m] \vdash Y(X_1\ldots X_m/x_1\ldots x_m)$ is not equivalent, because the left-hand side can be v-improper while the right-hand side v-proper by constructing a degenerated function that is undefined for all its arguments. To illustrate the loss of analytic information, consider two redexes $[\lambda x[^0+\ x\ ^01]\ ^03]$ and $[\lambda y[^0+\ ^03\ y]\ ^01]$. They both β-reduce to $[^0+\ ^03\ ^01]$. In the resulting Composition we lost the track of which function has been applied to which argument. As a solution we propose the rule of ***β-reduction by value*** that is valid and applicable in any context. Let $x_i \rightarrow_v \alpha_i$ be mutually distinct variables and let $D_i \rightarrow_v \alpha_i (1 \leq i \leq m)$ be constructions. Then the following rule is valid:

$$[[\lambda x_1\ldots x_m Y]D_1\ldots D_m] \vdash {}^2[^0Sub\ [^0Tr_{\alpha 1}D_1]^0x_1\ \ldots\ [^0Sub\ [^0Tr_{\alpha_m}D_m]^0x_m\ ^0Y]]$$

Example. "John loves his own wife. So does the Mayor of Ostrava."

$\lambda w\lambda t\ [\lambda x\ [^0Love_{wt}\ x\ [^0Wife\_of_{wt}\ x]]\ ^0\text{John}] =_{\beta v}$
$\lambda w\lambda t\ {}^2[^0Sub\ ^{00}John\ ^0x\ ^0[^0Love_{wt}\ x\ [^0Wife\_of_{wt}\ x]]]$

[7] For details see [3].

$\lambda w\lambda t\ [so\_does_{wt}\ {}^0MO_{wt}] \rightarrow$
$\lambda w\lambda t\ {}^2[{}^0Sub\ {}^0[\lambda w\lambda t\lambda x[{}^0Love_{wt}\ x[{}^0Wife\_of_{wt}\ x]]]\ {}^0so\_does\ {}^0[so\_does_{wt}\ {}^0MO_{wt}]] =_{\beta v}$
$\lambda w\lambda t\ [\lambda x\ [{}^0Love_{wt}\ x\ [{}^0Wife\_of_{wt}\ x]]\ {}^0MO_{wt}] =_{\beta v}$
$\lambda w\lambda t\ {}^2[{}^0Sub\ [{}^0Tr^{00}MO_{wt}]\ {}^0x\ {}^0[{}^0Love_{wt}\ x\ [{}^0Wife\_of_{wt}\ x]]].$

One can easily check that in all these construction whether reduced or non-reduced the track of the property of loving *one's own* wife is being kept. This property is constructed by the Closure $\lambda w\lambda t\lambda x\ [{}^0Love_{wt}\ x\ [{}^0Wife\_of_{wt}\ x]]$. When applied to John it does not turn into the property of loving John's wife. And the same property is substituted for the variable *so_does* into the second sentence. Thus we can easily infer that John and the Mayor of Ostrava share the property of loving their own wives.

4 Conclusion

We described generalization of Tichý's sequent calculus to the calculus for TIL 2010. The generalization concerns these issues. First, the extensional rules of existential generalization and substitution of identicals were generalized so that to be valid in any context, including intensional and hyperintensional ones. Second, we showed that the sequent calculus remains to be the calculus for TIL based on the ramified hierarchy of types with one important exception, which is the rule of β-reduction. We specified a generally valid rule of β-reduction 'by value' that does not yield a loss of analytic information about which function has been applied to which argument. No doubt that these are valuable results.

Yet some open problems and questions remain. Among them there are in particular the question on completeness of the calculus and the problem of its implementation.

Acknowledgements

The research reported herein was funded by Grant Agency of the Czech Republic Projects No. 401/10/0792, "Temporal Aspects of Knowledge and Information", 401/09/H007 'Logical Foundations of Semantics' and also by the internal grant agency of VSB-Technical University Ostrava, Project SP2012/26, "An utilization of artificial intelligence in knowledge mining from software processes".

References

1. Duží, M. (2012): Towards an extensional calculus of hyperintensions. *Organon F*, 19, supplementary issue 1, pp. 20–45.
2. Duží, M., Materna P. (2012): TIL jako procedurální logika. Průvodce zvídavého čtenáře Transparentní intensionální logikou.
3. Duží, M., Jespersen, B. (to apper), Procedural isomorphism, analytic information, and beta-conversion by value, forthcoming in *Logic Journal of the IGPL*, Oxford.

4. Duží, M., Číhalová, M., Ciprich, N., Frydrych, T., Menšík, M. (2009): Deductive reasoning using TIL. In RASLAN'09, Recent Advances in Slavonic Natural Langure Processing. Ed. Sojka, P., Horák, A.. Brno: Masarykova universita, pp. 25–38.
5. Duží, M., Jespersen B., Materna P. (2010): Procedural Semantics for Hyperintensional Logic; Foundations and Applications of Transparent Intensional Logic. Series Logic, Epistemology and the Unity of Science. Berlin, Heidelberg: Springer.
6. Tichý, P. (1982): Foundations of partial type theory. *Reports on Mathematical Logic*, 14: pp. 52–72. Reprinted in (Tichý 2004: pp. 467–480).
7. Tichý, P. (1986): Indiscernibility of identicals. *Studia Logica*, 45: pp. 251–273. Reprinted in (Tichý 2004: pp. 649–671).

Czech Knowledge-Based System with Temporal Reasoning

Andrej Gardoň

Faculty of Informatics, Masaryk University
Brno, Czech Repulic
xgardon@fi.muni.cz

Abstract. In this paper we discuss recent advancements in the field of knowledge representation and question-answering using natural language interfaces. The focus is to reveal projects' weak capabilities in temporal processing, namely time modeling and reasoning over temporal facts. With these deficiencies in mind, we propose new knowledge-based system called Dolphin suitable to parse sentences in Czech language, identify and model temporal information and infer answers to questions regarding time information. Fundamental theory behind Dolphin is Transparent Intensional Logic, which is high-order intensional logic calculus. Finally, we discuss Dolphin evaluation and future work.

Key words: question-answering, time representation, Dolphin

1 Introduction

Advancements in computer vision [1], artificial muscle fiber [2] and state-of-the-art systems including IBM Watson [3] have fundamentally influenced interaction between computers and humans. Projects like Watson, TrueKnowlege [4] or AURA [5]) aim at providing natural language interface to a user. They answer questions about factual or even scientific knowledge. Expansion of smart-phones, equipped with powerful processors, allows research in personal assistants that interact with the user and provide necessary information in real-time. Siri is an assistant for Apple phones and it focuses on question-answering in given domains. It waits for activation words and then uses so called active ontologies to answer the question. Siri even tells jokes [6]! Google's huge database of information, stored within its search engine, provides factual answers on natural language inputs to Google Now assistant[1]. It can also find navigation guidelines. Evi[2] is based on TrueKnowledge technology and is especially built for everyday information retrieval including answering the request *"give me a list of restaurants in my area"*. Aforementioned projects prove the possibility to move from standard key-driven approach in information retrieval

[1] http://www.google.com/landing/now
[2] http://www.evi.com

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2012, pp. 43–50, 2012.

towards more human-like methods. While processing of natural language input is well established, processing of temporal aspect is almost new feature in the projects.

2 Time representation and temporal reasoning

Time is an inseparable part of the space. Both components form so called time-space, the World in which our lives take place. Almost any utterance in natural language regards time information either in explicit or implicit form. For search engines like Google, representing temporal information is not mandatory as seeking for factual information is nearly time independent. However, assistants like Siri or Evi must realize time dimension to become human-like secretary. It is surprising that the most advanced question-answering system Watson is only able to detect basic temporal expressions. Its processing engine detects TLink (defined in TimeML [7]) relation in the input sentence and candidate answers. That relation specifies the end points of a general temporal relationship. To eliminate candidates that are temporally incompatible with the question, Watson computes *before* feature of the clue to determine whether it is chronologically before the entity came into the existence [8]. TrueKnowledge defines Temporal partner as a fact that references other fact and makes assertion about its validity. Key element of temporal partner is a Timepoint, single moment on the time line. Timeperiod is a class of two time points and is equivalent to an interval with start and end points. To cover infinite and indefinite time periods there are special objects including *iafter=unknown point in the future*, *timezore=point in time infinitely ago*, *forever=infinite future* [9]. Aura is a project that aims to provide scientific knowledge to everyone by natural language interface to books. HaloBook is a next generation electronic book based on Aura technology. The reader can interactively read the text (self-explanatory tooltips), ask questions and get explanations or learn the content through individualized tutoring [10]. Besides these advanced features, AURA supports just basic event ordering and partitioning of knowledge into situations. A situation represents changes of facts during time. Ordering of situations is also possible [11]. Advanced model of time is available in Cyc ontology. It utilizes two abstractions of time. The first one is interval-based and defines relations between intervals thanks to the primitives: *before* and *simultaneousWith*. The second one is set-based and it can state axioms like *"people do not eat and sleep at the same time"*. Temporal projection axioms associate default periods of persistence with propositions. With their use, it is possible to infer the name of a person in the future [12]. ThoughtTreasure is a comprehensive platform for natural language processing and common sense reasoning. It can keep a calendar up-to-date based on natural language input. Timestamps are basic objects for time representation and their combinations form timestamps ranges. Durations define the length of a given interval and Extended timestamps ranges represents

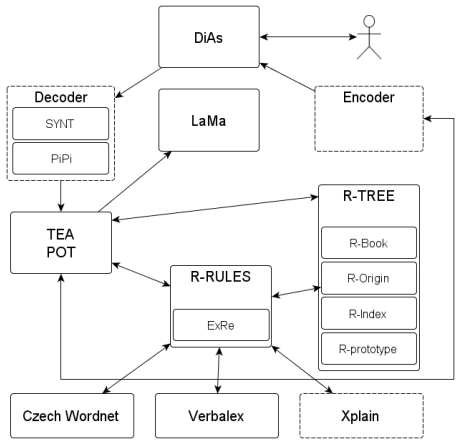


Fig. 1. Architecture of the Dolphin system

repeating or regularly scheduled events. ThouhtTreause looks to have the most advanced time model. On the other side it is ontology oriented. Set of specialized agents analyzes input texts and extract information that is encoded in the self-contained ontology. To utilize ThoughtTreasure on a new domain, the ontology has to be extended and agents must be added [13].

While temporal processing by knowledge systems is in early ages, support by formal languages is well established. TimeML is an example. This annotation language addresses time stamping of events, ordering of events, reasoning with underspecified temporal expressions (e.g. last week) and reasoning about the persistence of events. Events are defined as situations that happen or occur, and their detection is separated from identification of relations between them [7]. Interval temporal logic represents both propositional and first-order reasoning over period of time[3] . Transparent Intensional Logic (TIL) is a comprehensive high-order calculus that supports grammatical tenses, representation of events and episodes. It can process sentences like *"I go swimming every Friday"* and undefined time points and intervals like in *"I was there"*. Besides excellent temporal support, TIL can handle personal attitudes and incorporates the model of possible worlds. All these benefits make it good candidate for a knowledge-representation [14].

3 Dolphin system

Dolphin is a knowledge-based system that aims to process sentences in natural language, translate them into TIL formulas and provide reasoning capabilities to allow question-answering. Motivation for the project is to develop an

[3] `http://en.wikipedia.org/wiki/Interval_temporal_logic`

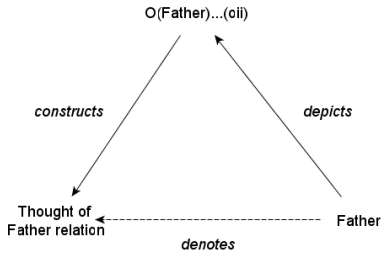


Fig. 2. Meaning of the Father relation

advanced knowledge-based system primarily understanding other language than English. In case of Dolphin, the mother language is Czech. As far as the authors known, there is no such system yet. Orientation on Czech is not limiting as architecture of the system is language independent. To support new language, one has to provide language specific decoder and encoder modules. Main goal of the project is advanced temporal processing. The system identifies most of temporal expressions in sentences and disposes by inference rules for their processing. Events and episodes are useful concepts from TIL that enable Dolphin to represent coherent view of subsequent situations and model relations between them. TIL supports deep analysis of grammatical tenses, undefined time intervals, repetitive activities and thus handles the most of time information in any sentence. Gardoň and Horák provide comprehensive introduction to temporal processing in Dolphin [15]. Following sections briefly describe essential modules from Dolphin's architecture depicted on Figure 1

- ***DiAs*** (Dialogue Assistant) handles the dialogue between a user and the system. It proposes questions to be asked and manages inputs/outputs from decoder and encoder modules.
- ***Decoder*** is a plug-in module that generally incorporates parser and PiPi (Post Processing). Input is the sentence in given language. After processing, there is a TIL construction on the output. Parser analyses the sentence to obtain syntactical, lexical, morphological and relational data from the sentence's parts. Based on the analysis, the parser translates the sentence into TIL construction. PiPi has to evaluate additional information from the sentence and prepare final output. For example, it utilizes anaphora resolution tool. In case of a question, it transforms simple TIL construction into TIL match that orders subsequent modules to treat the input as a query. In case of ambiguous parsing, decoder informs subsequent module about alternative inputs. For the purposes of Dolphin, we have utilized Synt parser that processes Czech sentences [16] and Saar a system for automatic anaphora resolution [17].
- ***LaMa*** (Language Manager) is a key component responsible for language independency of the system. TIL constructions encodes crucial information

from the sentence in procedure-like fashion. For example *"John is a father of Andrew"* is represented as a procedure Father over arguments John and Andrew. LaMa maps any procedure or argument on a referent's ID. TIL is based on the three-fold expression-meaning relation as depicted on the Figure 2. The picture represents meaning of the relation *Father* that is mapped on corresponding TIL construction (sense). The construction is an abstract procedure that construct a referent. Dolphin system approximates every referent in object-oriented manner, and it assigns universal ID to such approximation that is called Referent.

- ***TEA POT*** is heart of the system. TEA (Til Execution Agent) first invokes LaMa to translate TIL construction into R-format (words are substituted for referent IDs). The result of TEA is named R-TIL (referential TIL). TEA passes the processing of R-TIL to POT (Processing and Organizing Tool) that parses the construction and identifies its components (sub constructions). POT utilizes R-TREE and R-RULES to propose a response on the input that is send to Encoder module.
- ***R-TREE*** is the knowledge-base. It stores R-TIL constructions and provides methods for their manipulation. R-TIL Construction is kind of a function with exactly one result. Thus every R-TIL construction stores referenced R-TIL construction. Knowledge is organized into possible worlds, fundamental concept of TIL. It allows one fact to have different truthfulness depending on the considered world. It also makes it possible to process dialogue with a user in a separate world and reflect new knowledge only after final checking proved the consistence with the general world. Knowledge-base incorporates four components:
  - *R-Book* – basic evidence of registered Referents. It allows answering questions like *"Is every object that you know red?"*
  - *R-Origin* – a list of R-TIL constructions that referent particular R-TIL construction. This component tracks every natural language representation that denotes given Referent.
  - *R-Index* – indexing component that provides rapid search.
  - *R-Statistics* – Automatic prototype generation. It can be turned on/off for every R-TIL construction. Constructions representing a set have prototype generation initially activated. Prototype is a statistical member of a set. It represents the distribution of properties over legitimate set members. This allows to infer answer for question *"Can every bird fly?"*. With the knowledge that pigeon cannot fly, corresponding distribution of fly property for every bird is under 100%. Thus, Dolphin can generate answer *"Almost all birds can fly"*.
- ***ROOT*** is brain of the system. It consists of the repository of inference rules and ExRe (External Resources) component. These rules can be invoked to prove a statement, find an answer for a question, keep the consistency of the R-TREE or generate new knowledge. Besides rules in R-TIL format (e.g. R-TIL representation of if-then statements), ROOT contains direct C++ functions (C-rules) mapped on particular Referents. C-rules allow Dolphin to be interactive with outside components or boost the performance of

heavy-load inference rules treated in R-TIL-way. ExRe component is an API for external resources. Dolphin system employs Czech WordNet and Verbalex[4] as essential resources of common-sense. When the system needs additional information about Referent (not found in R-TREE), it queries ExRe component to possible obtain it. In future, interactive games like X-plain[5] can be used to ask a user for required knowledge.

– ***Encoder*** plug-in module that transforms R-TIL constructions back to natural language sentences.

4 Evalution and future work

Dolphin system is the highlight of author's thesis. In three years horizon, the goal is to implement essential Dolphin modules namely LaMa, TEA POT, R-TREE and ROOT. Previous research and prototype implementations of Dolphin system [18], [19], and [20] already contain parts of aformentioned modules and are valuable sources of information for the planned work. We expect the ROOT component to contain inference rules for temporal processing and ExRe implementations for Czech WordNet and Verbalex. To evaluate the system capabilities we plan to build a corpus of condensed news articles. Lidovky.cz or Tyden.cz provides summarizations at the top of each article. These summarizations are ideal start texts for the corpus. Basic Czech decoder (based on Synt and Saara) and manual refinement will transform corpus knowledge into internal R-TIL representation. This combination will also handle mainly time oriented questions contained in the corpus. These questions will demonstrate the system abilities to handle undefined time intervals, repetitive actions and contiguous events grouped in episodes. Output will be presented in clear and readable form (Encoder is not a part of current effort). In future, we hope to complete the Decoder for Czech to automatically parse any input. With the finalization of Encoder and Dias, Dolphin will become full-value member of the question-answering community. Our long-time goal is to develop Siri like assistant oriented on time management and planning.

5 Conclusions

This paper provides brief overview of current advancements in question-answering systems. It identifies a gap in their temporal abilities and proposes the architecture of Dolphin system that should enhance time processing by handling undefined intervals, repetitive actions and contiguous events. Being the long time effort, author reveals his intentions in project's implementation and uncovers further plans.

[4] `http://nlp.fi.muni.cz/cs/VerbaLex`

[5] `http://nlp.fi.muni.cz/projekty/x-plain/rules.php`

Acknowledgments

This work has been partly supported by the Czech Science Foundation under the project P401/10/0792.

References

1. Connolly, C.: A new integrated robot vision system from fanuc robotics. The Industrial Robot **34**(2) (2007) 103–106
2. Shahinpoor, M., Kim, K., Mojarrad, M.: Artificial Muscles: Applications of Advanced Polymeric Nanocomposites. Taylor & Francis (2007)
3. Ferrucci, D.A.: Introduction to "This is Watson". IBM Journal of Research and Development **56**(3/4) (2012)
4. Tunstall-Pedoe, W.: True Knowledge: Open-domain question answering using structured knowledge and inference. AI Magazine **31**(3) (2010) 80–92
5. Gunning, D., et al.: Project Halo update—progress toward digital Aristotle. AI Magazine **31**(3) (2010) 33–58
6. Aron, J.: How innovative is apple's new voice assistant, Siri? New Scientist **212**(2836) (2011) 24 –
7. Pustejovsky, J., et al.: The Specification Language TimeML. In Mani, I., Pustejovsky, J., Gaizauskas, R., eds.: The Language of Time: A Reader. Oxford University Press (2004)
8. Kalyanpur, A., et al.: Structured data and inference in deepqa. IBM Journal of Research and Development **56**(3/4) (2012)
9. Tunstall-Pedoe, W.: Knowledge storage and retrieval system and method (2006)
10. Gunning, D.: Halobook and progress towards digital aristotle. In: Innovative Applications of Artificial Intelligence, San Francisco, CA (2011) Invited talk.
11. Clark, P., Porter, B.: Km - situations, simulations, and possible worlds. Technical report, AI Lab, Univ Texas at Austin (1999)
12. Lenat, D.B., et al.: Cyc: toward programs with common sense. Commun. ACM **33**(8) (1990) 30–49
13. Mueller, E.T.: Natural Language Processing with Thought Treasure. Signiform (1998)
14. Horák, A.: The Normal Translation Algorithm in Transparent Intensional Logic for Czech. PhD thesis, Masaryk University (2002)
15. Gardoň, A., Horák, A.: Time dimension in the dolphin nick knowledge base using transparent intensional logic. In Habernal, I., Matoušek, V., eds.: Proceedings of the 14th international conference on Text, speech and dialogue. Lecture Notes in Computer Science / Lecture Notes in Artificial Intelligence, Springer (2011)
16. Horák, A.: Computer Processing of Czech Syntax and Semantics. Librix.eu, Brno, Czech Republic (1998)
17. Němčík, V.: The Saara framework: An anaphora resolution system for czech. In: RASLAN 2009: Recent Advances in Slavonic Natural Language Processing, Karlova Studánka, Czech Republic, Masaryk University, Brno, Czech Republic (2009) 49–54
18. Gardoň, A., Horák, A.: The learning and question answering modes in the dolphin system for the transparent intensional logic. In: RASLAN 2007 : Recent Advances in Slavonic Natural Language Processing, Karlova Studánka, Czech Republic, Masaryk University, Brno (2007) 29–36

19. Gardoň, A.: Dotazování s časovými informacemi nad znalostmi v transparentní intenzionální logice (in slovak) (2010)
20. Gardoň, A., Horák, A.: Knowledge base for transparent intensional logic and its use in automated daily news retrieval and answering machine. In: 3rd International Conference on Machine Learning and Computing (ICMLC 2011), Singapore, IEEE (2011) 59–63

Linguistic Logical Analysis of Direct Speech

Aleš Horák, Miloš Jakubíček, Vojtěch Kovář

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

Abstract. Logical analysis of natural language allows to extract semantic relations that are not revealed for standard full text search methods. Intensional logic systems, such as the Transparent Intensional Logic (TIL), can rigorously describe even the higher-order relations between the speaker and the content or meaning of the discourse.
In this paper, we concentrate on the mechanism of logical analysis of direct and indirect discourse by means of TIL. We explicate the procedure within the Normal Translation Algorithm (NTA) for Transparent Intensional Logic (TIL), which covers the language analysis on the syntactic and semantic levels. Particular examples in the text are presented in syntactically complicated free-word-order language, viz the Czech language.

Key words: direct speech, indirect speech, Transparent Intensional Logic, TIL, Normal Translation Algorithm, NTA, logical analysis, syntactic analysis, parsing

1 Introduction

The analysis of natural language texts on morphological and syntactic levels already achieved application level quality, for the mainstream languages [1]. On the other hand, the analysis of various aspects on the semantic level is still on the way to quality knowledge analysis and extraction (see e.g. [2] or other SemEval 2012 task results). Standard data mining and search techniques have already reached the top of their potential and researchers and knowledge engineers employ semantics in the natural language processing [3,4,5,6]. Most current practical systems that need to utilize knowledge representation of natural language in formal logic usually do not go beyond the scope of first-order logic, even though in the language, there is a number of higher-order phenomena such as belief attitudes, grammatical tenses or intensionality, all of which cannot be addressed properly within the first-order logic.

In the following text, we are dealing with logical analysis of natural language (NL) using the formalism of the Transparent Intensional Logic (TIL, [7]), an expressive higher-order logical system introduced by Pavel Tichý [8,9], which works with a complex hierarchy of types, temporal system of possible worlds and an inference system in development.

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2012, pp. 51–59, 2012.

The current work is a part of a long-term project aimed at providing norms for the "translation" of various NL phenomena to logical constructions, the Normal Translation Algorithm (NTA) [10,11]. The actual implementation of the system works on top of the Czech syntactic parser `synt` [12]. `Synt` is based on the robust meta-grammar formalism including context-free chart parsing enhanced with contextual actions for phrase and sentence level tests. The parser uses a meta-grammar of about 250 meta-rules for the description of the whole Czech language with automatic grammar expansion to technical parsing rules.

In the following text, we focus on the issues of analysis of complex sentences including direct discourse. We first discuss the formal definition of direct and indirect speech and their logical consequences. Then we explain in detail how the logical analysis in the `synt` parser works and how the syntactic and logical representation of direct speech is obtained. In Section 4, we describe the process of obtaining a corpus containing texts with direct speech, that was used extensively for studying various aspects of this language phenomenon and for evaluation of the parsing procedure.

2 Direct and Indirect Discourse

Direct and indirect forms of speech are related kinds of so called *reported speech*, i.e. those utterances, where the speaker refers to another utterance or utterances [13]. In the *direct speech* form, the (current) speaker uses an exact quotation of the original speaker:

> Waiter said: "Are you ready to order, sir?"
> Mr Smith replied: "Yes. I'll have the beef stew for starters and my wife would like tomato soup."

The corresponding indirect speech can look like:

> The waiter asked, whether Mr Smith was ready to order.
> He replied, that he would have the beef stew for starters and his wife would like tomato soup.

The main difference in the logical consequences lies in the change of the actual speaker positions in the reported clause. In case of the direct speech, the subject position is occupied by the original speaker and all speech aspects are related to him or her. On the other hand, the indirect form is completely related to the reporting speaker and all original speech aspects are *transformed* to this new subject. Especially, this results in higher usage of anaphoric expressions and thus higher level of ambiguity in the indirect form of reported speech.

3 Analysis of Direct Speech

In this section, we first describe the implementation of the syntactic and logical analysis in the `synt` parser and then concentrate on the additions specific to the analysis of direct speech sentences.

3.1 The Synt Parser

`Synt` is a rule-based parser designed specifically for morphologically-rich free-word-order languages and currently used mainly for Czech.[1] It operates by means of a modified head-driven chart analysis with a context-free backbone interconnected with predefined in-programmed (so Turing complete) contextual actions. The contextual actions are used to capture contextual phenomena like grammatical agreement or advanced (possibly non-local) linguistic features.

The underlying meta-rules are defined in the form of a meta-grammar consisting of about 250 rules. Each rule can be attached a precedence level, a list of actions and a derivation type. The precedence level makes it possible to include mutually exclusive rules into the grammar. The backbone rules are generated from a meta-rule during the process of automatic generation of full grammar from the meta-grammar according to the derivation type (e.g. permutation of all right-hand side non-terminals, enclitics checks, etc.).

The main result of the syntactic parsing procedure is an ordered set of constituent parsing trees that is potentially very big but zipped within a *shared packed forest* structure [14] provided with a fast algorithm for extracting *n* best trees according to a combined ranking function.

Each of these trees can be used as an input to another set of contextual actions that transform the tree to a logical formula in the TIL formalism, using lexical type and valency information extracted from the VerbaLex verb valency lexicon [15].

3.2 Syntactic Analysis of Direct Speech

A sentence with direct speech consists of the *direct speech* segment and a *reporting clause*. We analyze the reporting clause as the head element of the whole sentence, as the direct speech part often plays the role of subject or object in the reporting clause.[2] The structure of the direct speech part can be arbitrarily complex – it can consist of one or more sentences, or of an incomplete sentence. Therefore, we analyze the content of the direct speech by the *direct_speech* non-terminal that can cover one, or more, clauses, and also expressions, where the verb is not present.

Here comes the question, what should be actually considered a sentence in the context of direct speech. One segment of direct speech with one respective reporting clause can contain an arbitrary number of sentences, or even paragraphs, so it is often not clear where the sentence boundary should be. There are two straightforward approaches:

– Consider the whole pair, i.e. complex direct speech and the respective reporting clause, as one sentence.

[1] The `synt` grammar was also adapted for the Slovak and English languages, which are subject of further development.

[2] As in e.g.: *"Go away," he said.*

– Split the direct speech to multiple sentences and consider only the sentence closest to the reporting clause as its completion.

The first solution may seem more correct, because there is an immediate relationship between the reporting clause and all parts of the direct speech; however, it would lead to sentences consisting of thousands of words or even more. Parsing such sentences would be computationally unfeasible, therefore we do not consider it a good solution. Since all the respective relations are extra-syntactic (anaphoric relations, relative tenses, ...), we have developed a combined solution – complex sentences can be contained in one direct speech segment only in case the whole is not too long, otherwise, the direct speech is split at sentence boundaries and the rest of the direct speech analysis is linked to the reporting clause via a specific link used during the logical analysis phase. Such solution is best realizable from the technical point of view and it is also closest to what the currently available sentence segmenters do.

Having the sentence unit fixed, there are three possible combinations of where the reporting clause can be placed, with regard to the direct speech segment:

– The reporting clause comes before the direct speech segment – e.g. *He asked: "Would you bring me a beer?"*
– The reporting clause comes after the direct speech segment – e.g. *"Would you bring me a beer?" he asked.*
– The direct speech segment is divided into two parts, with the reporting clause between them – e.g. *"Would you," he asked, "bring me a beer?"*

Therefore, three basic rules are needed to address these three combinations:

```
clause   →   clause   ':'   direct_speech
clause   →   direct_speech    clause
clause   →   direct_speech    clause    ','    direct_speech
```

As mentioned above, the direct speech segment then rewrites to a complex sentence in quotes. In case the content of the direct speech cannot be analyzed by the *sentence* non-terminal, we allow the direct speech to rewrite as an arbitrary sequence of characters. These two analyses are mutually exclusive, since the `non_sentence` rule of `direct_speech` is analysed on a higher (i.e. less probable) rule level and is thus pruned away in the case where both `direct_speech` rules match.

```
  direct_speech   →   '"'   sentence   '"'
9:direct_speech   →   '"'   non_sentence   '"'
  non_sentence    →   /[^"]+/
```

One problem arises in the case where the direct speech is interrupted by the reporting clause, but it forms one logical unit, e.g. in the sentence shown above: *"Would you," he asked, "bring me a beer?"*. For example, the manual for annotators of the Prague Dependency Treebank [16] deals with this direct speech type by

using non-projective constituents.[3] In the `synt` parser, the intra-clause position of the reporting clause is analysed in a way similar to a *parenthesis*, i.e. a part of the original clause, which can be inserted between any two sentence constituents.

3.3 Logical Analysis of Direct Speech

The analysis of direct speech is not so loaded with the anaphora resolution problem as the indirect speech form, however, we can encounter situations, where the actual content of the direct speech clause is logically less related or even completely irrelevant. Let us have a look at the following examples

Peter said: "Hand me the book." (1)

Peter asked: "Hand me the ..." (2)

Peter thought: "The unicorn!" (3)

Peter screamed: "Aaaargh!" (4)

The example sentence (1) forms the standard reporting utterance with the two parts of reporting clause and direct speech reported clause. However, all the remaining examples fail on the syntactic level to be analysed as a (complete) clause. The sentence (2) contains an incomplete (probably interrupted) reported clause, sentence (3) shows, that Peter's thought is related with and individual object, and last, the sentence (4) represents an example of an object, which cannot be analysed even on the morphological level and stays here for a non-verbal sound.

The logical analysis of direct speech sentences in `synt` is related to the procedure of analysis of complex sentences, see [17]. The construction generated by this procedure for the sentence (1) can look like:[4]

$$
\begin{aligned}
&\lambda w_1 \lambda t_2 \Big[\mathbf{P}_{t_2}, \Big[\mathbf{Onc}_{w_1}, \lambda w_3 \lambda t_4 (\exists x_5)(\exists c_6)(\exists i_7) \Big(\\
&\quad \big[\mathbf{Does}_{w_3 t_4}, i_7, [\mathbf{Perf}_{w_3}, x_5] \big] \ \wedge \\
&\quad \wedge [\mathbf{Peter}_{w_3 t_4}, i_7] \ \wedge \ x_5 = [\mathbf{say}, c_6]_{w_3} \ \wedge \\
&\quad \wedge c_6 = {}^{0}\Big[\lambda w_8 \lambda t_9 (\exists x_{10})(\exists i_{11}) \Big(\big[\mathbf{Does}_{w_8 t_9}, Ty, [\mathbf{Perf}_{w_8}, x_{10}] \big] \ \wedge \\
&\qquad \wedge x_{10} = [\mathbf{hand\_sb\_st}, Já, i_{11}]_{w_8} \ \wedge \ [\mathbf{book}_{w_8 t_9}, i_{11}] \Big) \Big] \\
&\quad \Big) \Big], \mathbf{Anytime} \Big] \ldots \pi \\
&\mathsf{Peter}/(o\iota)_{\tau\omega}; \mathsf{say}/((o(o\pi)(o\pi))_{\omega} * n); \mathsf{hand\_sb\_st}/((o(o\pi)(o\pi))_{\omega} \iota); \\
&\mathsf{book}/(o\iota)_{\tau\omega};
\end{aligned}
\tag{5}
$$

[3] See [16, Section 3.6.1]

[4] The `synt` outputs are translated from the Czech language for the demonstration purpose here in the paper.

Type	*Id*	*Word/Phrase*	*Reference*
sentence	sent1	Peter said: "Hand me the book."	
clause	m1	Peter said	
np	m2	Peter	
clause	m3	_ Hand me the book	
pron_pers_zero	m_zerosubj1	_	
pron_pers_strong	m4	me	m2
np	m5	book	

Fig. 1. The Saara system – anaphora resolution of the example sentence[6] ("mN" in the table refers to the term "markableN.")

As we may see, the verbal object x_5 in this construction is connected with an argument of the higher-order type $*_n$ representing a (trivialized) construction of order n. In this case the construction c_6 generates a *proposition* (of type π, or $o_{\tau\omega}$), which keeps all the properties related to the meaning of Peter's speech.

The corresponding anaphoric expressions that connect the reporting and reported speech can be identified using the automatic anaphora resolution tool Saara [18] that works in relation to the `synt` syntactic parser. An example of the anaphoric links from Saara can be seen in Figure 1. This allows us to link the variable *Já* from the construction (5) with the subject variable i_7 there.

The appropriate type of the verb say is obtained during the logical analysis of lexical items in `synt` by means of consulting the VerbaLex verb valency lexicon [19]. The entry related to the verb *říct* (*say*) is presented in Figure 2. Each verb frame participant is labelled with a two-level semantic role, which can be used for specific information regarding the TIL type of each lexical item. Currently, the verb arguments denoted by the 1st-level role COM[7] are analysed as the TIL higher-order type $*_n$.

The analysis of the other three example sentences (2), (3) and (4) does not contain two clauses, as the reported part fails to form a (whole) clause. In the case of the sentence (3), the reported part could be analysed as a noun phrase denoting and individual concept, but the sentences (2) and (4) even do not provide any such characteristics. In such cases, the analysis does not analyse the (incomplete) content of the direct speech part and the resulting construction related the reporting verb only to the (individual) expression in the form of a

[6] Again, the sentence words are translated from Czech in which the tools operate.
[7] "something that is communicated by or to or between people or groups"

Fig. 2. VerbaLex entry related to the verb *říct* (*say*).

string of characters. For example, the sentence (4) thus receives the analysis:

$$\lambda w_1 \lambda t_2 \Big[\mathbf{P}_{t_2}, \Big[\mathbf{Onc}_{w_1}, \lambda w_3 \lambda t_4 (\exists x_5)(\exists c_6)(\exists i_7) \Big(\\ \big[\mathbf{Does}_{w_3 t_4}, i_7, [\mathbf{Perf}_{w_3}, x_5] \big] \ \wedge \\ \wedge \, [\mathbf{Peter}_{w_3 t_4}, i_7] \ \wedge \ x_5 = [\mathbf{scream}, c_6]_{w_3} \ \wedge \\ \wedge \, c_6 = {}^{0\iota}\text{“Aaaargh”} \\ \Big) \Big], \mathbf{Anytime} \Big] \ldots \pi \\ \text{Peter}/(o\iota)_{\tau\omega}; \text{scream}/((o(o\pi)(o\pi))_\omega *_n); \text{“Aaaargh”}/\iota \tag{6}$$

Due to the “polymorphic” nature of the higher-order type $*_n$ the type of the verbal object can accept the argument in any of the cases of the direct form.

4 Direct Speech Corpus

In order to study the issues of syntactic and logical representation a corpus of direct speech of about 20,000 sentences has been created. It is obvious that the definition of direct speech is quite broad and allows speculative interpretations

as to what should be considered as direct speech. To be able to build the corpus automatically we therefore restrained ourselves only to direct speech which is introduced and finished by quotes. We used the czTenTen corpus from which we selected candidate sentences using the Corpus Query Language (CQL [20]).

Obviously, the formulation of the CQL query was subject to a trade off between precision and recall. After numerous trials following query was concluded:

```
<s/> containing
(<s>
[word!="\""]* [k!="k1"] "\"" [word!="\""]+
[k="k5"] [word!="\""]+ "\"" [word!="\""]*
</s>)
```

The resulting corpus was then used as a testbed for the study of syntactic and logical properties of the direct speech form in common texts.

5 Conclusions

We have described an efficient conversion of Czech sentences with direct speech into logical formulae in the formalism of Transparent Intensional Logic (TIL), as a part of the Normal Translation Algorithm project. We have described the parser used, the process of syntactic analysis and creation of the logical formulae from the constituent syntactic trees. We have also described a corpus of Czech direct speech which has been newly created for purposes of studying the phenomenon and for evaluation.

The speed and the precision of the whole process is sufficient and promises its future usage in automatic reasoning and intelligent question answering. In the future, we will mainly concentrate on exploiting these results in real-word applications, which mainly means integrating the information gained from logical analysis into the complex pipeline of linguistic processing, including anaphora resolution or inter-sentence relationship analysis.

Acknowledgements

This work has been partly supported by the Ministry of Education of CR within the LINDAT-Clarin project LM2010013, by EC FP7 project ICT-248307 and by the Czech Science Foundation under the project P401/10/0792.

References

1. Matsuzaki, T., Tsujii, J.: Comparative parser performance analysis across grammar frameworks through automatic tree conversion using synchronous grammars. In: Proceedings of the 22nd International Conference on Computational Linguistics. (2008)

2. Specia, L., Jauhar, S., Mihalcea, R.: Semeval-2012 task 1: English lexical simplification. In: Proceedings of the 6th International Workshop on Semantic Evaluation (SemEval 2012), Montreal, Canada. (2012)
3. d'Amato, C., Fanizzi, N., Fazzinga, B., Gottlob, G., Lukasiewicz, T.: Ontology-based semantic search on the web and its combination with the power of inductive reasoning. Annals of Mathematics and Artificial Intelligence (2011) 1–39
4. Hoxha, J., Junghans, M., Agarwal, S.: Enabling semantic analysis of user browsing patterns in the web of data. arXiv preprint arXiv:1204.2713 (2012)
5. Christensen, J., Soderland, S., Etzioni, O., et al.: An analysis of open information extraction based on semantic role labeling. In: Proceedings of the sixth international conference on Knowledge capture, ACM (2011) 113–120
6. Efrati, A.: With semantic search, google eyes competitors. The Wall Street Journal (March 15, 2012)
7. Duží, M., Jespersen, B., Materna, P.: Procedural Semantics for Hyperintensional Logic. Foundations and Applications of Transparent Intensional Logic. Volume 17 of Logic, Epistemology and the Unity of Science. Springer, Berlin (2010)
8. Tichý, P.: The Foundations of Frege's Logic. de Gruyter, Berlin, New York (1988)
9. Tichý, P.: Collected Papers in Logic and Philosophy. Prague: Filosofia, Czech Academy of Sciences, and Dunedin: University of Otago Press (2004)
10. Horák, A.: The Normal Translation Algorithm in Transparent Intensional Logic for Czech. PhD thesis, Masaryk University, Brno (2002)
11. Horák, A.: Computer Processing of Czech Syntax and Semantics. Librix.eu, Brno, Czech Republic (2008)
12. Horák, A., Kadlec, V.: New Meta-grammar Constructs in Czech Language Parser synt. In: Lecture Notes in Artificial Intelligence, Proceedings of Text, Speech and Dialogue 2005, Karlovy Vary, Czech Republic, Springer-Verlag (2005) 85–92
13. Coulmas, F.: Direct and indirect speech. Volume 31. De Gruyter Mouton (1986)
14. Kadlec, V.: Syntactic analysis of natural languages based on context-free grammar backbone. PhD thesis, Masaryk University (2008)
15. Horák, A., Pala, K.: Building a large lexicon of complex valency frames. In: Proceedings of the FRAME 2007: Building Frame Semantics Resources for Scandinavian and Baltic Languages, Lund University, Sweden, Tartu, Estonia (2007) 31–38
16. Hajič, J., Panevová, J., Buráňová, E., Urešová, Z., Štěpánek, J., Pajas, P., Kárník, J.: Anotace na analytické rovině – Návod pro anotátory (2005) `http://ufal.mff.cuni.cz/pdt2.0/doc/manuals/cz/a-layer`.
17. Horák, A., Jakubíček, M., Kovář, V.: Analyzing time-related clauses in transparent intensional logic. In: Proceedings of Recent Advances in Slavonic Natural Language Processing 2011, Brno, Czech Republlic, Masaryk University (2011) 3–9
18. Němčík, V.: The Saara Framework: An Anaphora Resolution System for Czech. In: Proceedings of Recent Advances in Slavonic Natural Language Processing 2009, Brno, Czech Republlic, Masaryk University (2009) 49–54
19. Hlaváčková, D., Horák, A., Kadlec, V.: Exploitation of the VerbaLex Verb Valency Lexicon in the Syntactic Analysis of Czech. In: Proceedings of Text, Speech and Dialogue 2006, Brno, Czech Republic, Springer-Verlag (2006) 79–85
20. Jakubíček, M., Rychlý, P., Kilgarriff, A., McCarthy, D.: Fast Syntactic Searching in Very Large Corpora for Many Languages. In: PACLIC 24 Proceedings of the 24th Pacific Asia Conference on Language, Information and Computation, Tokyo (2010) 741–747

Building Evaluation Dataset for Textual Entailment in Czech

Zuzana Nevěřilová

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

Abstract. Recognizing textual entailment (RTE) is a subfield of natural language processing (NLP). Currently several RTE systems exist in which some of the subtasks are language independent but some are not. Moreover, large datasets for evaluation are prepared almost exclusively for English language.
In this paper we describe methods for obtaining test dataset for RTE in Czech. We have used methods for extracting facts from texts based on corpus templates as well as syntactic parser. Moreover, we have used reading comprehension tests for children and students. The main contribution of this article is the classification of "difficulty levels" for particular RTE questions.

Key words: textual entailment

1 Introduction

Automatic reasoning systems are currently a promising application of Natural Language Processing (NLP). Since automatic natural language understanding (NLU) is a topic difficult to grasp and formalize scholars try to resolve sub-problems of it. Hence recognizing textual entailment (RTE) is a good application of NLU methods.

Basically RTE solves a yes/no question: whether a text T entails a hypothesis H. In most cases H is a sentence a T is a coherent text – a "story". *T entails H* if the meaning of H, as interpreted in the context of T, can be deduced from the meaning of T [1]. In this context *deduction* is not equal to logical deduction and has to be understood in a broader context. It is considered that systems with high precision on deciding the RTE question "understand" a text in natural language. Apart from being a good evaluation measure RTE can aim for several applications such as intelligent searching or automatic summarization.

However, large resources for testing RTE systems are needed. In this paper we describe the process of building a gold-standard for evaluation of a RTE system. Currently a RTE system is being developed and its preliminary results were promising. However, we cannot evaluate further improvements if have no testing data.

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2012, pp. 61–66, 2012.

In section 2 we describe the state-of-the-art both in developing RTE systems and creating test data sets. Section 3 presents several methods for creating test data sets from corpus and other resources as well. We present and discuss the "difficulty levels" of RTE and their evaluation.

2 State-of-the-art

Recognizing textual entailment represents an important domain of research. Since 2004 RTE Pascal challenges started with manually annotated datasets of texts and hypotheses (H-T pairs) that covered seven different tasks:

- information retrieval
- comparable documents
- reading comprehension
- question answering
- information extraction
- machine translation
- paraphrase acquisition

In each of the subsets the annotators either generated hypotheses or identified H-T pairs from texts. Afterwards, annotators decided whether given text entailed the hypothesis or not. Thus the datasets contain both positive and negative examples. Moreover annotators were asked to replace anaphora by their appropriate references so that the RTE task would not concern anaphora resolution [3].

Pascal Challenges took place from 2004 to present time (last challenge was RTE-7 in 2011) and the datasets are available. The data is stored in an XML format describing `pairs` and their sub-elements: `text` and `hypothesis`. We adopted this format for our new resource of Czech H-T pairs.

Recent RTE systems use different techniques how to decide whether T entails H. Apart from the ad-hoc and shallow approaches the sound approaches (e.g. [11]) use

- tree transformation operations that generate the hypothesis from the given text
- knowledge based operations

Tree transformation operations concern computing tree edit distance (insertion, deletion, substitution) as well as rules for entailment and contradiction. For example replacing a token (word or word expression in the parse tree of the sentence) by its antonym leads (in most cases) in contradiction.

Knowledge based operations concern generalization using a knowledge base (e.g. WordNet [5] or dbPedia [9]) or antonymy processing. Missing knowledge is considered to be a bottleneck of RTE.

Representants of working systems are: BIUTEE[1], EDITS[2], VENSES[3] or Nutcracker[4].

3 Collection H-T pairs

RTE applications use several methods for automated entailment judgment. We have to reflect this fact when preparing RTE datasets. We also wanted to keep the information about extraction of the H-T pairs as well as the "difficulty level" of the entailment. The latter is not easy to obtain. However, we propose a classification of the pairs in the following subsection.

3.1 Reading comprehension tests for children/adults

We have analyzed reading comprehension tests for children and secondary school students. The classification reflects common reading comprehension problems w.r.t. reader's age.

- subsequence – the easiest entailment, most often it is a true entailment
- synonyms – replace a word in H by its synonym, obtain H' and then $H' \in T$, true entailment
- siblings – a word w_h in H is a sibling of a word w_t in T (w_h and w_t have common (direct) hypernym), but w_h and w_t are not synonyms, false entailment
- specification – a word w_h in H is a hyponym of a word w_t in T, false entailment
- syntactic rearrangement – H is a reformulation of a sentence in T, e.g. active–passive transformation or subordinate clause–object transformation
- interchanged arguments – all words from H are present in a sentence from T but their order or syntactic arrangement is different, false entailment
- qualifiers – the meaning of H is modified by a qualifier, judgment or by hedging
- anaphora – H is a paraphrase of a sentence s in T, but s contains anaphora and H contains reference, entailment value depends on anaphora resolution
- other – the meaning of H is not present in the context of T, other knowledge (encyclopedic, mathematical etc.) is needed or H is off-topic (and then the entailment is negative)

We have started with tests for 7-years-old children [10]. So far we have collected 12 documents with tests. We did a classification on 34 H-T pairs. Afterwards we have classified 24 H-T pairs extracted from secondary school leaving exam. Table 3.1 shows classification results. In tests for 7-years-old children

[1] http://u.cs.biu.ac.il/~nlp/downloads/biutee/protected-biutee.html
[2] http://edits.fbk.eu/
[3] http://project.cgm.unive.it/venses.html
[4] http://svn.ask.it.usyd.edu.au/trac/candc/wiki/nutcracker

Table 1. Classification of reading comprehension tests. Question types that are frequent in tests for 7-years-old children (left column) are expected to be easier to solve than questions frequent in tests for 18-years-old students.

question type	7-years	18-years
subsequence	20 %	0 %
synonyms	17 %	12 %
siblings	35 %	8 %
specification	2 %	4 %
syntactic rearrangement	5 %	50 %
interchanged arguments	5 %	3 %
qualifiers	0 %	17 %
anaphora	0 %	4 %
off-topic	16 %	21 %

each question is in one class, in final exam test several techniques are used at the same time (e.g. syntactic rearrangement together with hedging). Therefore the sum of the rightmost column is greater than 100 %. The classification was done by one annotator since it is a preliminary phase of the dataset development.

3.2 Corpus patterns

While observing questions in reading comprehension tests we have proposed several templates for extracting facts from corpora. Some parts of the templates are language independent while other are language dependent. We have used Corpus Query Language (CQL) in The Sketch Engine corpus tool [6]. This task is inspired by information extraction applications.

We were working with the Czech morphologically annotated and disambiguated corpus `czes` that contains 465,102,710 tokens[5].

Enumeration We have extracted nouns and adjectives following a noun in accusative with the column sign and delimited by commas and conjunctions a, nebo, ani (and, or, neither–nor). The hypothesis is then built rearranging the enumeration items, e.g. for a text "Každý objekt obsahuje tři základní datové komponenty: data, metody a atributy." (Each object contains three basic data components: data, methods and attributes.) we obtain three hypotheses such as "Metody jsou komponenty objektu." (Methods are components of the object.). This method extracted 738 hypotheses.

Passive We have extracted sentences with the verb *to be*, a passive verb and noun in instrumental. This is a typical passive construction in Czech and it is relatively easy to transform such sentences to active. We have

[5] 2012-06-21 size

obtained hypotheses such as "Lesy obklopují obec" (Forests surrond the village) from passive constructions such as "Obec je obklopena lesy." (The village is surrounded by forests). This method extracted 12.573 hypotheses.

Aliases We have extracted sentences containing "also known as". The hypothesis is created by stating that the alias is other name for an object, e.g. "Václava Zapletalová, jinak zvaná Wendy" (Vaclava Zapletalova, also known as Wendy) resulted to the hypothesis "Václavě Zapletalové se říká Wendy" (Wendy is a different name for Vaclava Zapletalova). This method extracted 26 hypotheses.

For sentence generation we used a system for Czech noun phrases declension [8]. This system is built upon the morphological analyser/generator `majka`.

In the last stage we are planning to use the tool `efa` for fact extraction [2]. It is based on syntactic parser SET [7] but moreover modules for recognizing information about time, location and manner are implemented.

3.3 Annotation

All these methods are used to extract H-T pairs from Czech texts. We plan to annotate each pair at least by two annotators independently. Moreover, in secondary school final exams correct answers are available. We expect high inter-annotator agreement in case of 7-years-old children and low inter-annotator agreement in case of secondary school final exam. In other methods we expect high coverage and high inter-annotator agreement since the methods are quite straightforward. According to [4] we plan to compute inter-annotator agreement. However, we plan to exclude H-T pairs where annotators will not agree on. The aim is to build a dataset with clear distinction what is a valid entailment and what is not.

4 Conclusion and Future Work

We have presented building an evaluation dataset for a system for recognizing textual entailment. We propose several resources of H-T pairs: reading comprehension tests, corpus querying using templates and fact extraction software. We have also presented an approach for judging the difficulty level of particular H-T pairs.

Future work concerns retrieval of more data from corpus. We will observe reading comprehension tests a create more patterns for paraphrasing sentences extracted from corpus.

In future we have to annotate the data by multiple annotators and to evaluate the inter-annotator agreement.

Acknowledgments

This work has been partly supported by the Czech Science Foundation under the project P401/10/0792 and by the Ministry of Education of CR within the LINDAT-Clarin project LM2010013.

References

1. Akhmatova, E.: Textual entailment resolution via atomic propositions. In: Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment (April 2005)
2. Baisa, V., Kovář, V.: Information extraction for czech based on syntactic analysis. In: Vetulani, Z. (ed.) Human Language Technologies as a Challenge for Computer Science and Linguistics, Proceedings of 5th Language and Technology Conference. pp. 466–470 (2011)
3. Challenges, P.: Recognising textual entailment challenge. online at http://pascallin.ecs.soton.ac.uk/
4. Dagan, I., Dolan, B., Magnini, B., Roth, D.: Recognizing textual entailment: Rational, evaluation and approaches. Natural Language Engineering 15(Special Issue 04), i–xvii (2009), http://dx.doi.org/10.1017/S1351324909990209
5. Fellbaum, C.: WordNet: An Electronic Lexical Database (Language, Speech, and Communication). The MIT Press (May 1998), published: Hardcover
6. 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
7. Kovář, V., Horák, A., Jakubíček, M.: Syntactic analysis using finite patterns: A new parsing system for czech. In: Human Language Technology. Challenges for Computer Science and Linguistics: 4th Language and Technology Conference, LTC 2009, Poznan, Poland, November 6-8, 2009. p. 161 (2011)
8. Neverilová, Z.: Declension of czech noun phrases. In: Actes du 31e Colloque International sur le Lexique et la Grammaire. pp. 134–138. České Budějovice (2012)
9. Orlandi, F., Passant, A.: Modelling provenance of DBpedia resources using wikipedia contributions. Web Semantics: Science, Services and Agents on the World Wide Web 9(2), 149 – 164 (2011), http://www.sciencedirect.com/science/article/pii/S1570826811000175, <ce:title>Provenance in the Semantic Web</ce:title>
10. Střední odborná škola Otrokovice, s.s.v.a.z.p.d.v.p.p.: Pracovní listy k nácviku porozumění čtenému textu a nápravě čtení. http://www.zkola.cz/zkedu/pedagogictipracovnici/kabinetpro1stupenzsamaterskeskoly/metodickematerialyvyukoveprogramy/pracovnilistykporozumenitextuanapravecteni/default.aspx (2003),
11. Stern, A., Dagan, I.: A confidence model for syntactically-motivated entailment proofs. In: Proceedings of the International Conference Recent Advances in Natural Language Processing 2011. pp. 455–462. RANLP 2011 Organising Committee, Hissar, Bulgaria (September 2011), http://www.aclweb.org/anthology/R11-1063

Part III

Text Corpora and Tools

Detecting Spam in Web Corpora

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. To increase the search result rank of a website, many fake websites full of generated or semigenerated texts have been made in last years. Since we do not want this garbage in our text corpora, this is a becoming problem. This paper describes generated texts observed in the recently crawled web corpora and proposes a new way to detect such unwanted contents. The main idea of the presented approach is based on comparing frequencies of n-grams of words from the potentially forged texts with n-grams of words from a trusted corpus. As a source of spam text, fake webpages concerning loans from an English web corpus as an example of data aimed to fool search engines were used. The results show this approach is able to detect properly certain kind of forged texts with accuracy reaching almost 70 %.

Key words: web corpora, spam detection

1 Introduction

Web spamming has become a well know problem on the Internet. Spammed web pages contain hyperlinks, nonsense or very low quality texts in order to skew search engine results. The aim is to bring Internet users' attention to these in fact irrelevant pages. Seen through the eyes of an Internet browsing person, web spamming results in unwanted or unrelated content.

Another problem caused by web spam is distortion of frequency of words in collections of texts gathered from the Internet. This research is aimed at proposing a new method for detecting such unwanted spam contents. Comparing recently created web corpora for English, we observe more spammed data in a more recent corpus.

For example, word *viagra* is approximately hundred times more frequent in a web corpus from 2012 than in its predecessor from 2008. EnTenTen12, the focus corpus, was gathered from the web in May 2012 and cleaned using boilerplate removal and 7-gram based deduplication algorithms [1]. Comparable cleaning techniques were applied to the older corpus, therefore the cleaning procedure was unable to deal with increased presence of the word *viagra*.

Tables 1 and 2 show lemmas of words which are significantly more frequent in the most recent web corpus than in older web corpora. Looking at the top

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2012, pp. 69–76, 2012.

	lemma	2012	2008	RFR
1	**loan**	360.1	51.7	6.97
2	online	462.4	119.2	3.88
3	your	4194.4	1660.2	2.53
4	**insurance**	263.1	56.8	4.63
5	**credit**	321.7	119.9	2.68
6	buy	421.3	175.7	2.40
7	**mortgage**	132.4	22.9	5.78
8	product	502.6	219.6	2.29
9	brand	164.3	41.8	3.93
10	website	261.9	94.5	2.77
...				
21	**debt**	150.9	48.5	3.11

Table 1. Keywords from focus corpus enTenTen12 (2012), reference corpus enTenTen08 (2008).

	lemma	2012	2008	RFR
1	**loan**	360.1	65.1	5.53
7	**credit**	321.7	106.3	3.03
20	**mortgage**	132.4	32.3	4.10
26	**debt**	150.9	46.7	3.23
112	**insurance**	263.1	157.1	1.67

Table 2. Keywords from focus corpus enTenTen12 (2012), reference corpus ukWaC (1996).

items, there is a suspiciously high amount of words from domain of money: *loan, insurance, credit, mortgage, debt, etc.* RFR stands for relative frequency ratio between columns 2008 and 2012. The keywords extraction [2] function of SketchEngine was used. There are frequencies per million (FPM) in appropriate corpora in columns 2012 and 2008. The keywords are sorted by *keyness rank* which is order of a lemma in the comparison according to keyness score

$$\frac{\text{FPM in focus corpus} + 100}{\text{FPM in reference corpus} + 100}.$$

Authors of [3] also claim the amount of web spam increased dramatically and define several types of spam techniques. This work is interested in the following types:

- dumping of a large number of unrelated terms,
- weaving of spam terms into copied contents,
- phrase stitching (gluing together sentences or phrases from different sources).

Cleaning procedures applied to enTenTen12 were not designed to remove that type of spam, therefore the aim of this work is to detect such spam texts,

especially phrase stitching. Other spam categories can be dealt with boilerplate removal tools (e.g. pages containing only hyperlinks) or deduplication tools (e.g. pages copying whole paragraphs from other sources).

Another big issue is a rapid growth of content farms (see the second item in Table 3) – low quality articles promoting goods, services or webpages also made just to increase a page rank. Although being sparsely informative and much repetitive, such text is syntactically and sematically correct. That is why we do not intend to remove it from text corpora and this work does not aim to detect such kind of web content.

2 Sources of spam data

For the purpose of evaluating our method, we selected two sources of generated data: web documents about loans and fake scientific articles. The data was obtained in November 2012 from the web. Boilerplate removal [1] was applied.

2.1 Recent web corpus

Since there is a whole group of *loan* (and generally money) related words among the top keywords in the comparison in Tables 1 and 2, we chose to study documents containing word *loan* in enTenTen12. 200 documents were randomly chosen and the source web pages displayed in a browser for examination. However, only 92 pages were successfuly downloaded – this work was done 6 months after crawling the pages, many of them were not found or contained a different text.

Table 3 shows classification of web pages in the collection. We classified 44.5 % of documents not suitable for a text corpus as a spam. We selected 406 paragraphs from random documents and evaluated them once again, since some paragraphs in spam documents were not spam and vice versa. Finally, 199 (49 %) spam and 207 (51 %) not spam paragraphs were used in further experiments.

text category	**class**	**% doc**
nice text	OK	37.0 %
low quality text (possibly a content farm)	OK	18.5 %
fluency slightly broken (sentence or paragraph stitching)	spam	9.8 %
fluency broken (sentence or phrase stitching)	spam	13.0 %
not fluent (triplets of words stitching)	spam	14.1 %
nice text, unrelated words (spam terms weaving)	spam	7.6 %

Table 3. Classification of texts from the collection of 92 web documents containing word *loan*. Texts not containing fluent paragraphs were marked as spam.

3 All n-grams approach to detect generated texts

N-grams are the most common resource for statistical language modeling. Language models are used in many areas, mainly in speech analysis and in machine translation. In the latter, a language model is responsible for a fluent translation.

It was shown that language models assign more probability to a fluent text than to a random sequence of words. In well-known evaluation method for quality of machine translation BLEU [4], n-grams (where n is from 1 to 4) are used for measuring fluency of a candidate sentence and this method correlates reasonably well with human evaluations.

Usually, n-grams with n up to 4 are used. It means that higher-order n-grams are not taken into account. There are several reasons why not to use higher-order n-grams: slower performance, higher hardware requirements and mainly sparse data.

In our method, we suppose that a fluent text can be achieved by using a language model as in machine translation. When generating non-sense text, a language model relies on n-grams up to some n. Let us suppose now $n = 2$. Since the model has information about bigram counts (and probabilities) but knows nothing about trigrams, we might recognize this fact simply by checking frequencies of trigrams from the generated text taken from a reference corpus. Generally, if a model used n-grams, we could always check n+1-grams.

In other words, we suppose that n-grams of an order higher than an order used in a model will have much lower frequencies since the model simply does not know about them. Using a trigram model, generated quadrigrams will be more or less random.

3.1 Frequencies of all n-grams in a corpus

Our method does not use standard probabilities as in usual language models. We use simple frequencies of all n-grams of all orders. For a given sentence, we check all unigrams, bigrams, trigrams, ... n-grams where n is sentence length.

To be able to do that, we need a reference corpora (we used British National Corpus, BNC) and a procedure which quickly gets counts of all possible n-grams. There are $O(m^2)$ of all possible n-grams in corpus with m word positions. We used algorithm described in [5], which produces these counts in $O(n)$ time.

The algorithm uses *suffix array* [6], *longest common prefix array* [7] and fact that majority of all n-grams are unique in a corpus. In Table 4 you can see a part of suffix array built from the BNC.

Since n-grams in the suffix array are sorted, we can observe, that there are at least six bigrams 'distinguish at', exactly three trigrams 'distinguish at least' and only one quadrigram 'distinguish at least between', 'distinguish at least four' etc. All n-grams starting with 'distinguish at least four' has frequency 1 and the algorithm exploits this observation.

distinguish	at	all	between	the	personal	...
distinguish	at	least	between	the	meaning	...
distinguish	at	least	four	sources	in	...
distinguish	at	least	three	cases	:	...
distinguish	at	once	from	the	destruction	...
distinguish	at	the	outset	between	the	...

Table 4. Part of suffix array built form BNC, n-grams starting with *distinguish.*

3.2 Frequency-drop between n-gram orders

For the classification we needed a list of features to be able to use a machine learning method. For that purpose we used frequency-drop between various levels of n-grams in a sentence. The first level (word counts) is compared with the second level (bigram counts) and so on. Counts are summed up for a level and these sums are simply divided between levels. A frequency-drop is ratio between a sum of n-grams frequencies from a level a and a sum of n-gram frequencies from a level $a + 1$. We suppose that the more fluent and natural a text is the smaller frequency-drops should be between its appropriate n-gram levels and that a significantly bigger frequency-drop is located between levels which correspond to an order of a language model used for generating the text.

On Figure 1 you can see an explaining diagram. Numbers bellow the words are frequencies of various n-grams. Frequencies on the first line with numbers correspond to unigrams. The second line contains frequencies for bigrams etc. On the right side, there are three frequency-drop numbers which stand for ratios between sums of appropriate n-grams levels.

Since the text is generated and it is not fluent English text, frequency-drops on various levels are very low (less than 0.01).

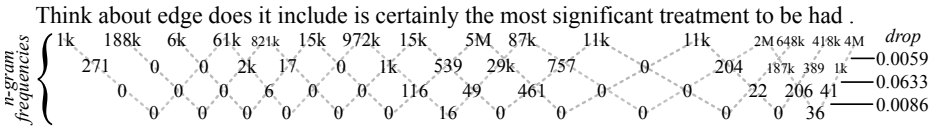


Fig. 1. Frequency-drop on 4 levels for apparently generated sentence.

Some low frequent words as e.g. proper names may introduce substantial frequency-drops between n-gram levels, but in one variant of our method we use average frequency-drops values, which should solve the problem especially for longer sentences.

On the contrary, some high frequent words may introduce the same from the other side, but this effect is weakened again by using an additional average value.

3.3 Classification

List of frequency-drops are vectors with values from 0 to 1. In our data maximum frequency-drop level with non-zero sum was 7 (i.e. there was an 8-gram with non-zero frequency). At first we used only frequency-drops from non-zero levels, but then added another value: average frequency-drop for all levels. In that case, accuracy was improved slightly. As another feature, paragraph length (in sentences) was added which also slightly improved the results.

For the classification we used both simple threshold tuning and Support Vector Machine (SVM) [8] for machine learning.

In the first experiment on development data, two thresholds were combined: frequency-drop between first two levels (0.015) and average frequency-drop on all levels (0.025).

For the SVM we used the mentioned n-tuples with two additional values – average frequency-drop and paragraph length. SVM automatically chose its kernel-function to maximalize accuracy of the trained model.

4 Results

For evaluation of these methods we used standard metrics: *precision, recall, accuracy* and *f-score*. In case of spam classification, it is usefull to express these metrics using terms *true positive* (*tp*), *true negative* (*tn*), *false positive* (*fp*) and *false negative* (*fn*). When a paragraph is annotated manually as 'spam' and classified by one of our methods as 'spam', then it is *true positive* match. The other matches are analogical. Standard metrics can be then expressed as follows.

$$precision = \frac{tp}{tp + fp}$$

$$recall = \frac{tp}{tp + fn}$$

$$accuracy = \frac{tp + tn}{tp + tn + fp + fn}$$

$$f\text{-}score = 2 \times \frac{precision \times recall}{precision + recall}$$

In Table 5 you can see results for the two methods. BASE is baseline method: classification of all paragraphs as 'spam'. sDEV is the simple threshold method run on development data, sTEST is the same method run on test data. In next columns, the SVM method with various training vectors are listed. SVM used n-tuples with up to 7 frequency-drop levels. SVM^c used one more value for a number of sentences in a paragraph, SVM_a used one more value for average frequency-drop and the last SVM_a^c used the two additional values together.

We can see that the simple threshold method is only slightly better than the baseline. So is the first SVM method using vector of frequency-drops for n-gram levels. The best method is SVM^c which gives almost 70 % accuracy.

	BASE	**sDEV**	**sTEST**	**SVM**	**SVM^c**	**SVM$_a$**	**SVM$_a^c$**
precision	48.53	55.36	49.72	31.31	83.84	72.73	84.85
recall	100.0	97.03	90.91	50.00	63.36	64.29	62.22
f-score	65.35	70.50	64.29	38.51	72.17	68.25	71.79
accuracy	48.53	59.41	50.98	51.47	68.63	67.16	67.65

Table 5. Results for various evaluated methods.

5 Conclusions and future work

Processing of test data was done on vertical files. Development and test sets have about 17,000 tokens respectively.

We classified spam on paragraph level. It might be more appropriate to classify spam on sentence level since as was said, sometimes only a part of a paragraph is generated and the rest is taken over from another web page. Moreover the mentioned deduplication tool removes duplicates on paragraph level so it can not solve the problem with sentences stitched together in a paragraph.

The accuracy of these proposed methods can be decreased also by imperfect training data. It was annotated manually by non-native speaker of English which probably influenced quality of the annotation. For further research we would like to use data annotated by more annotators and especially data annotated by native speakers.

The process of getting frequency-drops on all possible n-gram levels is very fast. After suffix array, longest commonest array and all n-gram frequencies are prepared, we are able to process cca 2,000,000 tokens per minute on a computer with 8 processors and 100 GB RAM. It allows us to process very large billion corpora within hours or days. Standard language modeling methods (using probabilities) would be much slower and thus unusable for big data.

In the future we would like to use n-grams extracted from larger corpora, e.g. from mentioned ententen08. As a reference corpus we used BNC. We would also like to try the method on bigger training and testing data for specific domain of scientific texts. For that purpose we intend to use fake scientific articles generated by Scigen[1]. As a counterpart to these fake articles, random documents published in Computer Science section in arXiv[2] could be used. For that data set we would use a corpus containing scientific documents.

Spam and other garbage on the web is increasing problem nowadays and we should try our best to be able to deal with it and filter it out. Without it, methods for machine translation, speech analysis etc. would be badly affected by low quality data used for building n-gram language models.

[1] An Automatic Computer Science Paper Generator, `http://pdos.csail.mit.edu/scigen/`

[2] Open access to natural sciences e-prints, `http://arxiv.org/list/cs/recent`

6 Acknowledgement

This work has been partially supported by the Ministry of Education of CR within the LINDAT-Clarin project LM2010013 and by EC FP7 project ICT-248307.

References

1. Pomikálek, J.: Removing Boilerplate and Duplicate Content from Web Corpora. PhD thesis, Masaryk University, Brno (2011)
2. Kilgarriff, A.: Getting to know your corpus, Springer (2012) 3–15
3. Gyongyi, Z., Garcia-Molina, H.: Web spam taxonomy. (2005)
4. Papineni, K., Roukos, S., Ward, T., Zhu, W.: Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting on association for computational linguistics, Association for Computational Linguistics (2002) 311–318
5. Yamamoto, M., Church, K.: Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus. Computational Linguistics **27**(1) (2001) 1–30
6. Manber, U., Myers, G.: Suffix arrays: a new method for on-line string searches. siam Journal on Computing **22**(5) (1993) 935–948
7. Kasai, T., Lee, G., Arimura, H., Arikawa, S., Park, K.: Linear-time longest-common-prefix computation in suffix arrays and its applications. In: Combinatorial Pattern Matching, Springer (2006) 181–192
8. Suykens, J., Vandewalle, J.: Least squares support vector machine classifiers. Neural processing letters **9**(3) (1999) 293–300

Recent Czech Web Corpora

Vít Suchomel

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

Abstract. This article introduces the largest Czech text corpus for language research – *czTenTen12* with 5.4 billion tokens. A brief comparison with other recent Czech corpora follows.

Key words: web corpora, Czech

1 Introduction

Algorithms in the field of natural language processing generally benefit from large language models. Many words and phrases occur rarely, therefore there is a need for very large text colletions to research behaviour of words. [1] Furthermore, the quality of the data obtained from the web is also stressed. [2] Language scientists are increasingly turning to the web as a source of language data. [3] Nowadays, the web is the biggest, easily exploitable and the cheapest source of text data.

We decided to support corpora based research of Czech language by building a new Czech corpus from web documents. The aim was to apply successful data cleaning tools and label the words with grammatical categories.

2 Building a new Czech web corpus

CzTenTen12 is a new Czech web corpus built in 2012 using data obtained from the web in 2011. Several automatic cleaning and postprocessing techniques were applied to the raw data to achieve a good quality corpus for language research.

2.1 Crawling the web

We used web crawler *SpiderLing*[1], our previous work [4], to gather the data from the web. We started the crawl from 20000 seed URLs spanning over 8600 domains. The URLs were chosen using Corpus Factory [5], Czech Wikipedia and partially from older web corpus *czTenTen*. The crawl was restricted to the Czech national top level domain (`.cz`). 15 million documents of size 500 GB were downloaded in 24 days.

[1] `http://nlp.fi.muni.cz/trac/spiderling`

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2012, pp. 77–83, 2012.

2.2 Postprocessing and tagging

The crawler performed character encoding detection and converted the data to UTF-8. The crawler detected language using character trigrams and filtered out texts in other languages than the focus language. Extra care had to be taken in case of Slovak which contains similar trigrams and unwanted texts may pass the filter. We prepared a list of Czech words not present in Slovak and a dual list of Slovak words not present in Czech. Using these lists, paragraphs containing three times more unique Slovak words than unique Czech words (0,4 %) or unique words mixed from both languages (6,4 %) were separated.

Boilerplate removal tool jusText[2] was used to remove html markup, page navigation, very short paragraphs and other useless web content. The data was de-duplicated by removing exact and near duplicate paragraphs using tool onion[3]. Paragraphs containing more than 50 % seen 7-grams were dropped.

Paragraphs containing only words without diacritical marks were tagged for further use, e.g. when studying the informal language of the web.[4] Parts containing more than 20 % of words not recognized by morphological analyzer *Desamb* were considered nonsense and removed from the corpus. The final size of the corpus reaches 5.4 billion tokens (4.4 billion words).

Czech morphological analyzer *Desamb* [6,7] was used to tag the corpus. The added information consists in the part of speech and other grammatical categories (where applicable): gender, number, case, aspect, modality and other.[5]

3 Comparison with other corpora

The following recent Czech corpora are used in the comparison:

- *SYN 2010* . . . Czech national corpus – the SYN-series corpora up to 2010[6][7],
- *czes2* (a web corpus from 2009),
- *czTenTen* (a web corpus from 2011),
- the Hector project corpus[8] (*Hector*) [2].

3.1 Basic properties

According to [2], both *SYN* and *Hector* are deliberately balanced, e.g. the latter consists of 450 millions of words from news and magazines, 1 billion of words

[2] `http://nlp.fi.muni.cz/projects/justext`

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

[4] These texts come mostly from discussions or other informal websites (where poeople do not bother writing proper accent marks).

[5] Reference of full tagset: `http://nlp.fi.muni.cz/projekty/ajka/tags.pdf`

[6] `http://ucnk.ff.cuni.cz/english`

[7] Although the full corpus is not publicly available, a wordlist with frequencies was enough to carry out measurements presented later on.

[8] `http://hector.ms.mff.cuni.cz`

Table 1. Basic comparison of corpora. Only words consisting of letters are accounted in the word count. *Dictionary size* is the number of unique words with at least 5 occurrences. The *the-score* is the rank of word "the" in a list of words sorted by frequency from the most frequent one. The lower the value, the higher contamination by foreign words should be expected.

corpus	word count [10^6]	dictionary size [10^6]	the-score
SYN2010	1300	1.61	7896
czes2	367	1.03	42
czTenTen	1652	2.42	1023
Hector	2650	2.81	1184
czTenTen12	4439	4.16	1223

Table 2. Corpus distance measured for each couple of corpora. The lower the *distance score*, the more similar is the couple.

corpus	czes	czTenTen	Hector	czTenTen12
SYN2010	1.60	1.70	2.28	1.73
czes2		1.44	2.16	1.52
czTenTen			1.79	1.12
Hector				1.65

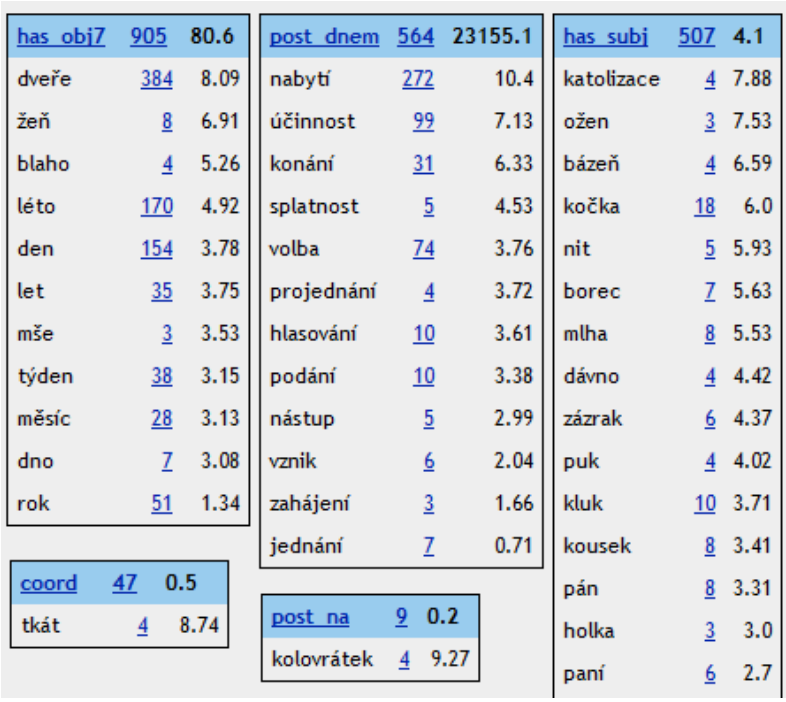
from blogs and 1.1 billion of words from discussions. The content of the new corpus was not controlled and a deeper analysis of content remains for further research.

Table 1 displays values of three metrics calculated for five corpora. We observe *czTenTen12* is the largest corpus with the largest dictionary. The *the-score* is a very simple metric offering a basic idea about contamination of the corpus by foreign (English) words. We observe *czes2* is the most polluted corpus and *SYN2010* is the most clean corpus in this measurement.

3.2 Corpora similarity

Table 2 shows a corpus comparison cross-table. The *distance score* calculation is based on relative corpus frequencies of 500 most frequent words in all corpora. The full method is described in [8]. We observe *czTenTen* and *czTenTen12* are very close. That can be explained by similar way of obtaining and processing the data and sharing a lot of documents. On the other hand, the balanced corpora are more distant.

Comparison of keywords (also based on the relative corpus frequency) in *czTenTen12* most different from *SYN2010* was published in [9]. We observe there are more discussions and blogs (informal words, verbs in 1st or 2nd person, pronouns, adverbs) and computer related words in the new unbalanced corpus. Comparing *czTenTen12* to *Hector*, we find the differrence in presence of informal words too. Top *czTenTen12* related words in this comparison are quite formal: *již*, *lze*, *oblasti*, *společnosti*, *zařízení*, *této*, *roce*, *zde*, *mohou*, *rámci*, *projektu*, *těchto*,



has_obj7	905	80.6
dveře	384	8.09
žeň	8	6.91
blaho	4	5.26
léto	170	4.92
den	154	3.78
let	35	3.75
mše	3	3.53
týden	38	3.15
měsíc	28	3.13
dno	7	3.08
rok	51	1.34

post_dnem	564	23155.1
nabytí	272	10.4
účinnost	99	7.13
konání	31	6.33
splatnost	5	4.53
volba	74	3.76
projednání	4	3.72
hlasování	10	3.61
podání	10	3.38
nástup	5	2.99
vznik	6	2.04
zahájení	3	1.66
jednání	7	0.71

has_subj	507	4.1
katolizace	4	7.88
ožen	3	7.53
bázeň	4	6.59
kočka	18	6.0
nit	5	5.93
borec	7	5.63
mlha	8	5.53
dávno	4	4.42
zázrak	6	4.37
puk	4	4.02
kluk	10	3.71
kousek	8	3.41
pán	8	3.31
holka	3	3.0
paní	6	2.7

coord	47	0.5
tkát	4	8.74

post_na	9	0.2
kolovrátek	4	9.27

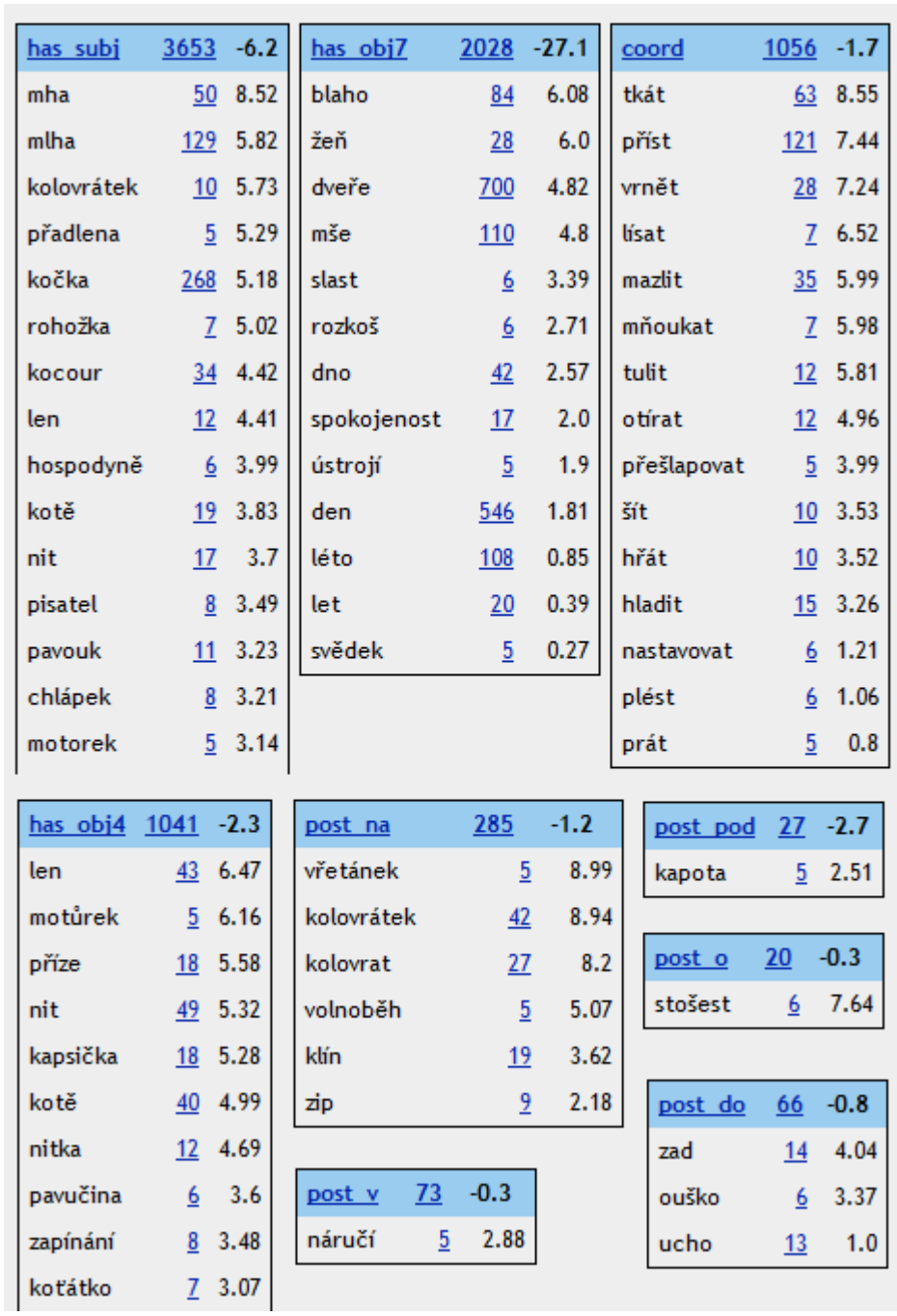
Fig. 1. Word sketch for word *přís*t in *czes2*. A part of grammatical relations is displayed. The number of hits of the word in the corpus is 5191.

systému. They could belong to some project notes or contracts. The key words of the opposite direction are *no*, *holky*, *jo*, *xD*, *D*, *blog*, *teda*, *taky*, *já*, *dneska*, *sem*, *jdu*, *máš*, which leads to conclusion *Hector* contains more blogs and discussions (generally informal texts) than *czTenTen12*.

3.3 Word sketches – bigger is better

Figures 1 and 2 display word sketch for word *přís*t in *czes2* and *czTenTen12* in SketchEngine[9]. As can be easily observed, the bigger corpus offers better words in relations with the head word. E.g. (*přís*t, *blaho*) in relation *has_obj7* (which stands for a verb with a noun in instrumental case) is a quite common collocation in Czech. That is well reflected in the sketch for *czTenTen12* with 84 occurrences and the first place by saliency score in the relation table. However, the smaller corpus offers only 4 instances of this collocation. Relation *has_obj4* (which stands for a verb with a noun in accusative case) in *czes2* is very poor, while containing many well suiting words in the case of the bigger corpus: *len*,

[9] `http://sketchengine.co.uk/`



has_subj	3653	-6.2
mha	50	8.52
mlha	129	5.82
kolovrátek	10	5.73
přadlena	5	5.29
kočka	268	5.18
rohožka	7	5.02
kocour	34	4.42
len	12	4.41
hospodyně	6	3.99
kotě	19	3.83
nit	17	3.7
pisatel	8	3.49
pavouk	11	3.23
chlápek	8	3.21
motorek	5	3.14

has_obj7	2028	-27.1
blaho	84	6.08
žeň	28	6.0
dveře	700	4.82
mše	110	4.8
slast	6	3.39
rozkoš	6	2.71
dno	42	2.57
spokojenost	17	2.0
ústrojí	5	1.9
den	546	1.81
léto	108	0.85
let	20	0.39
svědek	5	0.27

coord	1056	-1.7
tkát	63	8.55
přist	121	7.44
vrnět	28	7.24
lísat	7	6.52
mazlit	35	5.99
mňoukat	7	5.98
tulit	12	5.81
otírat	12	4.96
přešlapovat	5	3.99
šít	10	3.53
hřát	10	3.52
hladit	15	3.26
nastavovat	6	1.21
plést	6	1.06
prát	5	0.8

has_obj4	1041	-2.3
len	43	6.47
motůrek	5	6.16
příze	18	5.58
nit	49	5.32
kapsička	18	5.28
kotě	40	4.99
nitka	12	4.69
pavučina	6	3.6
zapínání	8	3.48
koťátko	7	3.07

post_na	285	-1.2
vřetánek	5	8.99
kolovrátek	42	8.94
kolovrat	27	8.2
volnoběh	5	5.07
klín	19	3.62
zip	9	2.18

post_v	73	-0.3
náručí	5	2.88

post_pod	27	-2.7
kapota	5	2.51

post_o	20	-0.3
stošest	6	7.64

post_do	66	-0.8
zad	14	4.04
ouško	6	3.37
ucho	13	1.0

Fig. 2. Word sketch for word *přist* in *czTenTen12*. A part of grammatical relations is displayed. The number of hits of the word in the corpus is 28276, that is 5 times more frequent than in the smaller corpus.

příze, nit, nitka, pavučina.[10] Furthermore, most of collocations with prepositions in prepositional relations *post_na*, *post_v*, *post_do* and other are present just in the word sketch of the bigger corpus: *na kolovratu, na klíně, v náručí, do ouška, pod kapotou, ostošest*. We conclude a bigger corpus is much more useful for language research based on collocations of words.

4 Conclusion and future work

This article introduced the largest Czech text corpus for language research. A basic comparison with other contemporary Czech corpora was made. Example work sketches were shown to support idea that bigger corpora are better.

The future plans for building web corpora of Slavonic languages include gathering resources in Polish and Croatian. Another interesting research opportunity is studying semantic topics automatically extracted from documents in the corpus. That would help us to know more about the content of the corpus and consequently of the Czech web.

Acknowledgements

This work has been partially supported by the Ministry of Education of CR within the LINDAT-Clarin project LM2010013, by the Ministry of the Interior of CR within the project VF20102014003 and by the Czech Science Foundation under the project P401/10/0792.

References

1. Pomikálek, J., Rychlý, P., Kilgarriff, A.: Scaling to billion-plus word corpora. Advances in Computational Linguistics **41** (2009) 3–13
2. Spoustová, J., Spousta, M.: A high-quality web corpus of czech. In: Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12), Istanbul, Turkey, European Language Resources Association (ELRA) (may 2012)
3. Kilgarriff, A., Grefenstette, G.: Introduction to the special issue on the web as corpus. Computational linguistics **29**(3) (2003) 333–347
4. Suchomel, V., Pomikálek, J.: Efficient web crawling for large text corpora. In: Proceedings of the Seventh Web as Corpus Workshop, Lyon, France (2012)
5. Kilgarriff, A., Reddy, S., Pomikálek, J., Pvs, A.: A corpus factory for many languages. Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'10, Malta) (2010)
6. Šmerk, P.: Unsupervised Learning of Rules for Morphological Disambiguation. In: Lecture Notes in Artificial Intelligence 3206, Proceedings of Text, Speech and Dialogue 2004, Berlin, Springer-Verlag (2004) 211–216
7. Jakubíček, M., Horák, A., Kovář, V.: Mining phrases from syntactic analysis. In: Lecture Notes in Artificial Intelligence, Proceedings of Text, Speech and Dialogue 2009, Plzeň, Czech Republic, Springer-Verlag (2009) 124–130

[10] Presence of other words in this relation is caused by tagging mistakes or by putting them in a wrong relation.

8. Kilgarriff, A.: Comparing corpora. International journal of corpus linguistics **6**(1) (2001) 97–133
9. Kilgarriff, A.: Getting to know your corpus. In: Text, Speech and Dialogue, Springer (2012) 3–15

CzAccent – Simple Tool for Restoring Accents in Czech Texts

Pavel Rychlý

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

Abstract. There are many Czech text written without any accents. The paper describes a tool for fully automatic restoration of Czech accents. The system is based on a simple approach of big lexicon. The resulting accuracy of the system evaluated on large Czech corpora is quite high. The system is in regular use by hundreds of users from around the whole world.

Key words: diacritic restoration, Czech, CzAccent

1 Introduction

The written form of the Czech language uses the same 26 character as English many of them with several different accents. The list of all accented characters of Czech is in Table 1. In the early days of personal computers in 80s of the last century and in the early days of mobile phones in the beginning of this century, the devices was to prepared for easily writing of accented characters and users wrote Czech texts without accents. Even today, there are people who write some texts or all texts without accents.

For most people, reading Czech text without accents is harder than reading correct texts. Many non-accented words are ambiguous, there are more than one possible way how to add one or more accents to create different words. On the other hand, all native speakers do not have problems with understanding the non-accented text, they are able to add correct accents to words in a context.

Table 1. All accented characters in Czech.

á	í	ť	Á	Í	Ť
č	ň	ú	Č	Ň	Ú
ď	ó	ů	Ď	Ó	Ů
é	ř	ý	É	Ř	Ý
ě	š	ž	Ě	Š	Ž

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2012, pp. 85–89, 2012.

Table 2. Relative frequency of accented characters in Czech texts compared to respective non-accented ones.

character	% in text	character	% in text
a	6.861	r	4.002
á	2.073	ř	1.129
c	2.543	s	4.617
č	0.962	š	0.779
d	3.659	t	5.583
d′	0.024	ť	0.043
e	7.996	u	3.165
é	1.077	ú	0.142
ě	1.424	ů	0.496
i	4.617	y	1.732
í	2.896	ý	0.855
n	6.517	z	2.020
ň	0.066	ž	0.972
o	8.146		
ó	0.030		

There was several attempts to build an automated tool for adding accents, usually based on learning n-grams of characters. The presented system outperform all character based systems.

The structure of the pager is following: next section states the complexity of the problem, then the CzAccent system is described in details. Next two sections provide results of evaluation and usage options of the system.

2 Complexity of Restoring Czech Accents

Accented vowels are very common in Czech texts, many other accented characters are very rare. The relative frequency of accented characters together with respective non-accented variant are listed in Table 2.

We can see that most frequent accented characters *á* and *í* are also most frequent accents compared to respective non-accented characters. The *á* character occurs in 23 % of all *a* occurrences (accented or non-accented). The *í* character occurs in almost 40 % of all *i* occurrences.

3 CzAccent method

The CzAccent system is based on a big lexicon. The the all words known to the Czech morphology analyser Majka was looked in a big corpus. The most frequent accented word from all possible accented words and also the original non-accented word was selected and added to the CzAccent lexicon. In the

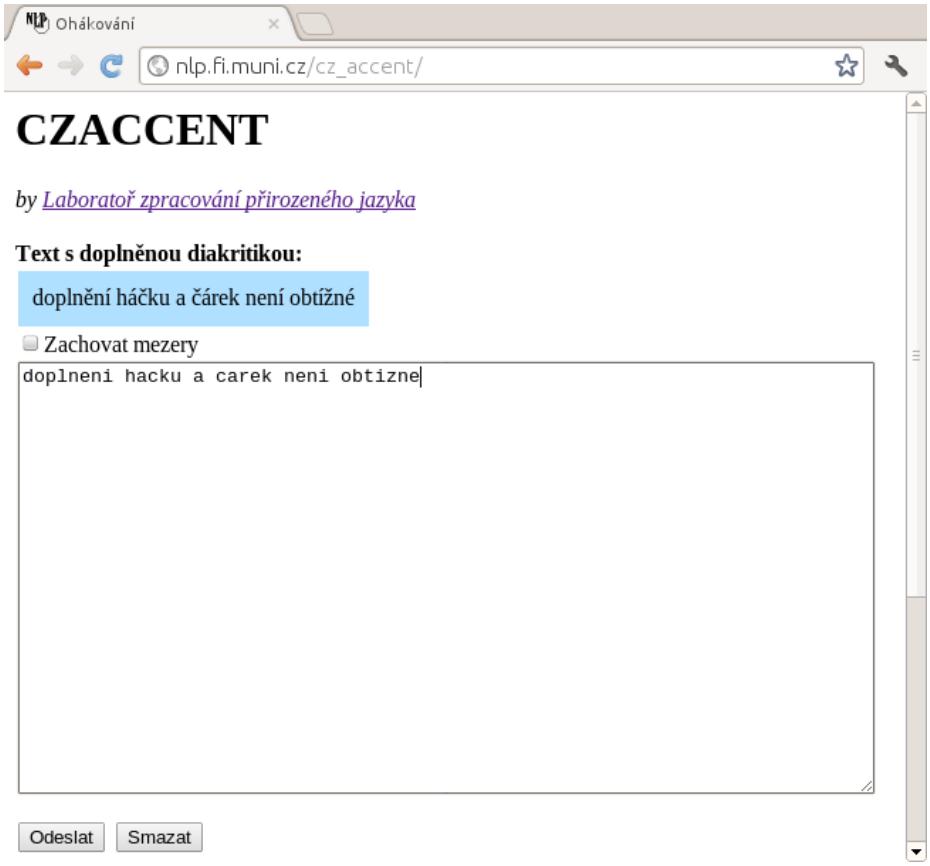


Fig. 1. CzAccent web interface.

result, there are millions of words which are stored in a very compact data structure using a finite state automaton [1]. The final data file containing the whole lexicon is very small, it has only 1.86 kB.

The CzAccent system processes any text in a straightforward way, it repeats the following steps from the beginning to the end of an input.

1. read a word (sequence of alphabetical characters),
2. try to find the word in the lexicon,
3. if found print the accented variant,
4. else print original word,
5. copy any non-alphabetical characters from input to output.

Due to its simplicity, the system is very fast. It can process more than 4.7 MB per second on moderate machine.

4 Evaluation

The system was evaluated on big Czech corpus CZES. CZES was built purely from electronic sources by mostly automated scripts and systems. [2]
Texts in the CZES corpus come from three different sources:

1. automated harvesting of newspapers (either electronic version of paper ones or electronic only), with annotation of publishing dates, authors and domain; these information is usually hard to find automatically from other sources;
2. customised processing of electronic versions of Czech books available online; and
3. general crawling of the Web.

The whole corpus should contain Czech texts only. There are small parts (paragraphs) in Slovak or English because they are parts of the Czech texts. Some Czech newspapers regularly publish Slovak articles, but we have used an automatic method to identify such articles and remove them from the corpus.

There was no restriction on the publication date of texts. There are both latest articles from current newspapers and 80 year old books present in the corpus.

The second corpus for evaluation was 1 million word corpus DESAM [3]. It is manually annotated and that is the reason that it is also very clean.

The accuracy of the system on the CZES corpus is 92.9, the accuracy on DESAM is 97.3. We can see that on cleaned texts the accuracy is very high.

5 Interface

The system is stable, it can be run in the form of a command line tool. An example of usage is at Figure 2.

```
$ echo realny problem | czaccent
reálný problém
```

Fig. 2. An example of CzAccent command line tool usage.

There is also a public web interface at the following address: `http://nlp.fi.muni.cz/cz_accent/`. It is in the form a simple page with one entry field. A user can enter a Czech text without accents and the system provides accented text. A screen-shot of the web interface is at Figure 1.

6 Conclusions

The system uses very simple method, but the resulting accuracy of the system evaluated on very large Czech corpus is quite high. The system is in regular use by hundreds of users from around the whole world.

Acknowledgements

This work has been partly supported by the Ministry of Education of CR within the LINDAT-Clarin project LM2010013 and by EC FP7 project ICT-248307.

References

1. Daciuk, J.: Finite state tools for natural language processing. In: Proceedings of the COLING 2000 workshop Using Toolsets and Architectures to Build NLP Systems, Luxembourg. (2000)
2. Horák, A., Rychlý, P.: Discovering grammatical relations in czech sentences. (2009)
3. Pala, K., Rychlý, P., Smrž, P.: DESAM – Annotated Corpus for Czech. In: Proceedings of SOFSEM '97, Springer-Verlag (1997) 523–530

Towards 100M Morphologically Annotated Corpus of Tajik

Gulshan Dovudov, Vít Suchomel, Pavel Šmerk

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

Abstract.
The paper presents a work in progress: building morphologically annotated corpus of Tajik language of the size more than 100 million tokens. The corpus is and will be by far the largest available computer corpus of Tajik: even its current size is almost 85 million tokens. Because the available text sources are rather scarce, to achieve the goal also the texts of a lower quality have to be included. This short paper briefly reviews the current state of the corpus and analyzer, discusses problems with either "normalization" or at least categorization of low quality texts and finally also the perspectives for the nearest future.

Key words: web corpora, Tajik

1 Introduction

The Tajik language is a variant of the Persian language spoken mainly in Tajikistan and also in some few other parts of Central Asia. Unlike closely related Iranian Persian which uses Arabic script, Tajik is written in Cyrillic.

Since the Tajik language internet society (and consequently the potential market) is rather small and Tajikistan itself is ranked among developing countries, available tools and resources for computational processing of Tajik as well as publication in the field are rather scarce. Availability of Tajik corpus data does not seem to change during the last year: aside from our corpus, the biggest freely available corpus is still the one within the Leipzig Corpora Collection project [5] (ca. 100 000 sentences, 1.8 million words, huge amount of errors), the biggest planned corpus is still the one prepared by the Tajik Academy of Sciences (target size 10 million, no visible changes in the last year)[1]. For further details and for information on other minor corpus projects see our previous work [1].

In this paper we present our corpus of contemporary Tajik language of more than 85 million tokens. After a brief review of its current state we will discuss problems with low quality texts. Finally we will debate possible improvements in the nearest future.

[1] http://www.termcom.tj/index.php?menu=bases&page=index3&lang=eng (in Russian)

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2012, pp. 91–94, 2012.

2 The Current State of the Corpus

Our corpus is built only from online sources (or, to be precise, only from sources which were online at some time, as the accessibility of some data changes rapidly). We use two different approaches to obtain the data. For the details refer to our previous papers [1] and [2].

The main part of the corpus was collected by crawling several portals, mostly news portals, in Tajik language.[2] Each portal is processed separately to get the maximum of relevant (meta)information, i.e. correct headings, publication date, rubric etc. The data for the second part of the corpus was obtained with SpiderLing, a general web crawler for text corpora [6], which automatically walks through the internet and searches for texts of a particular language. The crawling process started with 2570 seed URLs (from 475 distinct domains) collected with Corpus Factory [3]. The obtained data was uniformely tokenized and then deduplicated using Onion[3] with moderately strict settings[4].

The actual size of the corpus is 84,557,502 tokens and 70,665,499 words i.e. tokens which contain only Cyrillic characters (the rest is punctuation, numbers, Latin words etc.). The semi-automatically crawled part has 57,636,441 tokens, which means that the contribution of the automatically crawled part is 26,921,061 tokens. Our morphological analyzer recognizes 92.5 % of words (i.e. of the above mentioned 70 million tokens).

The corpus is not freely available for a download at the moment, but eventual interested researchers can access it through a very powerful corpus manager the Sketch Engine[5] [4].

2.1 Dealing with texts of lower quality

As was mentioned in the Introduction, Tajik uses Cyrillic script. Unfortunately, the Tajik alphabet contains six letters which are missing in the probably most widespread codepage in the post-Soviet world, i.e. cp1251. As there is or has been almost no support for Tajik in Windows and also in other major OSs, in many occasions people writing Tajik texts were not able to write proper Tajik-specific characters and were about to use some replacements. The most frequently used replacement sets can be seen in the table.[6]

Note that letters Љљ, Њњ, Її, and Ўў are shared by the Replacement sets 1 and 2, but except for Ўў the replacements have different meanings. Thus it would not be correct to directly replace all these replacement characters, but

[2] Paradoxically, the two largest Tajik news portals are not located in Tajikistan, but in the Czech Republic (ozodi.org, Tajik version of Radio Free Europe/Radio Liberty) and the United Kingdom (bbc.co.uk, Tajik version of BBC).

[3] http://code.google.com/p/onion/

[4] Paragraphs with more than 50 % of duplicate 7-tuples of words were removed.

[5] http://ske.fi.muni.cz/open/

[6] Note that Ss is not Latin S, but "Cyrillic Dze", Ii is not Latin I, but "Cyrillic Byelorussian-Ukrainian I" and the accents above Ќќ Ѝѝ should be acute, not grave (probably wrong glyph in LaTeXfont).

Table 1. Replacement sets for Tajik.

Char	RS1	RS2	RS3	RS4
Ғғ	Ѓѓ	Љљ	Uu	Гг
Ӣӣ	Її	Њњ	Bb	Ии
Ҷҷ	Љљ	Її	Xx	Чч
Ҳҳ	Њњ	Іі	{[	Хх
Ққ	Ќќ	Ss	Rr	Кк
Ӯӯ	Ўў	Ўў	Ee	Уу

one have to guess the whole replacement set which the particular document uses. Moreover, sometimes people write й instead of -и and bigrams х, к, or ч, (i.e. letter and comma) instead of proper letters хқч.

Out of 192,664 documents (web pages, newspapers articles etc.), 169,233 need not any change (87.8 %), 21,524 needs some replacements, in 1323 documents the "diacritics" was restored (RS4), and finally for 584 there was need both for replacements and the diacritics restoration.

2391 documents use RS1, 778 RS3 and 113 documents use RS2. The bigrams were used in 2453 words and й instead of -и in 77812 words. In total, 859641 words was somehow modified which is more than 1 % of all words in the corpus.

Numbers of particular changes and other useful information are described in each document's metadata which allows users to create specific subcorpora, e.g. subcorpus of texts without any changes (probably the most quality ones).

3 Future Work

The semi-automatic crawling was run a year ago, then after four months and then now. The automatic crawling was run a year ago, then after four months, after six months, and finally now. From the respective tables it can be seen, that the growth is not strictly linear, but the achievement of 100 million tokens seems to be possible during the next year.

Table 2. Growth of the semi-automatically crawled part.

date	tokens	increment	per month
11/2011	40.6 M	—	—
03/2012	43.3 M	+6.7 %	1.7 %
11/2012	47.2 M	+9.1 %	1.1 %

In the nearest future we want to further improve the analyzer to achieve better coverage and also we want to employ some kind of morphological guessing of unknown words. Then we will be able to develop a tagger and some tools for complex verbs and noun phrases detection which all will allow

Table 3. Growth of the automatically crawled part.

date	tokens	increment	per month
11/2011	34.6 M	—	—
03/2012	41.1 M	+18.6 %	4.7 %
05/2012	44.7 M	+8.6 %	4.3 %
11/2012	54.8 M	+22.6 %	3.8 %

us to create word sketches [4] for Tajik words. That is why we need to have the corpus as big as possible: 100 million words is considered as a minimum for word sketches to work reasonably.

Acknowledgements

This work has been partly supported by Erasmus Mundus Action II lot 9: Partnerships with Third Country higher education institutions and scholarships for mobility, and by the Ministry of Education of CR within the LINDAT-Clarin project LM2010013.

References

1. Dovudov, G., Pomikálek, J., Suchomel, V., Šmerk, P.: Building a 50M Corpus of Tajik Language. In: Proceedings of the Fifth Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2011. Masaryk University, Brno (2011)
2. Dovudov, G., Suchomel, V., Šmerk, P.: POS Annotated 50M Corpus of Tajik Language. In: Proceedings of the Workshop on Language Technology for Normalisation of Less-Resourced Languages (SALTMIL 8/AfLaT 2012). European Language Resources Association (ELRA), Istanbul (2012)
3. Kilgarriff, A., Reddy, S., Pomikálek, J., PVS, A.: A Corpus Factory for Many Languages. In: Proceedings of the Seventh International Conference on Language Resources and Evaluation, LREC 2010. Valleta, Malta (2010)
4. Kilgarriff, A., Rychlý, P., Smrž, P., Tugwell, D.: The Sketch Engine. In: Proceedings of EURALEX. pp. 105–116 (2004)
5. Quasthoff, U., Richter, M., Biemann, C.: Corpus Portal for Search in Monolingual Corpora. In: Proceedings of the Fifth International Conference on Language Resources and Evaluation, LREC 2006. Genoa (2006)
6. Suchomel, V., Pomikálek, J.: Practical Web Crawling for Text Corpora. In: Proceedings of the Fifth Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2011. Masaryk University, Brno (2011)

Part IV

Language Modelling

Building A Thesaurus Using LDA-Frames

Jiří Materna

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

Abstract. In this paper we present a new method for measuring semantic relatedness of lexical units, which can be used to generate a thesaurus automatically. The method is based on a comparison of probability distributions of semantic frames generated using the LDA-frames algorithm. The idea is evaluated by measuring the overlap of WordNet synsets and generated semantic clusters. The results show that the method outperforms another automatic approach used in the Sketch Engine project.

Key words: thesaurus, LDA-frames, semantic relatedness, lexical semantics

1 Introduction

Identifying meaning of words is one of the crucial problems in linguistics. While ordinary monolingual dictionaries index words alphabetically and provide a definition for every record, thesauri index words senses and group words with similar meaning. However, there is an important difference between lexicons of synonyms and thesauri. The clustered words in thesauri are not exactly synonyms. Thesauri rather group words with similar patterns of usage or semantically related words across parts of speech. As in other areas of linguistics, there is an important issue of polysemy. Since most of words in natural languages may have different meanings depending on the contexts in which they are used, a word usually belongs to multiple clusters. For instance, the word *bank* has two meanings – a financial institution and a border of a river, thus it should belong into two clusters.

Thesaurus is not only a useful resource helping to find and understand related words or phrases, which is mainly used by writers when hesitating what word they should choose. A word cluster or ranked list of similar words has many applications in natural language processing. One such application is the information retrieval task. In an information retrieval system, the query can be augmented by semantically related terms, which may lead to better retrieval quality.

One of the most popular English thesauri is Roget's thesaurus. It is a widely used English language thesaurus created by Dr. Peter Mark Roget in nineteenth century [6]. Another manually created resource grouping similar

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2012, pp. 97–103, 2012.

words together is WordNet [3]. WordNet is an electronic lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms called synsets, each expressing a distinct concept. Moreover, synsets are interlinked by means of conceptual-semantic and lexical relations. In comparison to Roget's thesaurus, which is primarily intended to be used by humans, WordNet is more often utilized in natural language processing taks.

Since the manual creation of thesauri, and the dictionaries in general, is a very time-consuming work, there are some attempts to create thesauri automatically by processing corpora. The similarity between words is usually measured by looking at their usage in texts. The same approach is used in a thesaurus generated using the Sketch Engine [7]. The similarity of lexical units in the Sketch Engine is measured by comparing so called word sketches. Word sketches are automatic, corpus-based summaries of a word's grammatical and collocational behaviour, which takes as input a corpus of any language and corresponding grammar patterns. The resulting summaries are produced in the form of a ranked list of common word realizations for a grammatical relation of a given target word.

In this work we proposed a similar method, which, instead of comparing word sketches, compares semantic frames of target words. Because the LDA-frames approach provides a probabilistic distribution over all frames, and is able to distinguish between different word senses, this method acquires better results than the Sketch Engine. It is demonstrated by measuring overlap with WordNet synsets.

2 LDA-frames

LDA-frames [8] is an unsupervised approach to identifying semantic frames from semantically unlabelled text corpora. There are many frame formalisms but most of them suffer from the problem that all frames must be created manually and the set of semantic roles must be predefined. The LDA-Frames approach, based on the Latent Dirichlet Allocation [1], avoids both these problems by employing statistics on a syntactically tagged corpus. The only information that must be given is a number of semantic frames and a number of semantic roles to be identified. This limitation, however, can be avoided by automatic estimation of both these parameters.

In the LDA-frames, a frame is represented as a tuple of semantic roles, each of them connected with a grammatical relation i.e. subject, object, modifier, etc. These frames are related to a predicate via a probability distribution. Every semantic role is represented as a probability distribution over its realizations.

The method of automatic identification of semantic frames is based on the probabilistic generative process. We treat each grammatical relation realization as generated from a given semantic frame according to the word distribution of the corresponding semantic role. Supposing the number of frames is given by the parameter F and the number of semantic roles by R. The realizations are

generated by the LDA-Frames algorithm as follows.

For each lexical unit $u \in \{1, 2, \ldots, U\}$:

1. Choose a frame distribution φ_u from $\mathrm{Dir}(\alpha)$.
2. For each lexical unit realization $t \in \{1, 2, \ldots, T\}$ choose a frame f_{ut} from $\mathrm{Mult}(\varphi_u)$, where $f_{ut} \in \{1, 2, \ldots, F\}$:
3. For each slot $s \in \{1, 2, \ldots, S\}$ of the frame f_{ut}
   (a) look up the corresponding semantic role r_{uts} from $\rho_{f_{ut}s}$, where $r_{uts} \in \{1, 2, \ldots, R\}$.
   (b) generate a grammatical realization w_{uts} from $\mathrm{Multinomial}(\theta_{r_{uts}})$

The graphical model for LDA-Frames is shown in the figure 1. In this model, $\rho_{f,s}$ is a projection $(f, s) \mapsto r$, which assigns a semantic role for each slot s of a frame f. This projection is global for all lexical units. The multinomial distribution of words, symbolized by θ_r, for a semantic role r is generated from $\mathrm{Dirichlet}(\beta)$. The model is parametrized by hyperparameters of prior distributions α and β, usually set by hand to a value between 0.01 – 0.1.

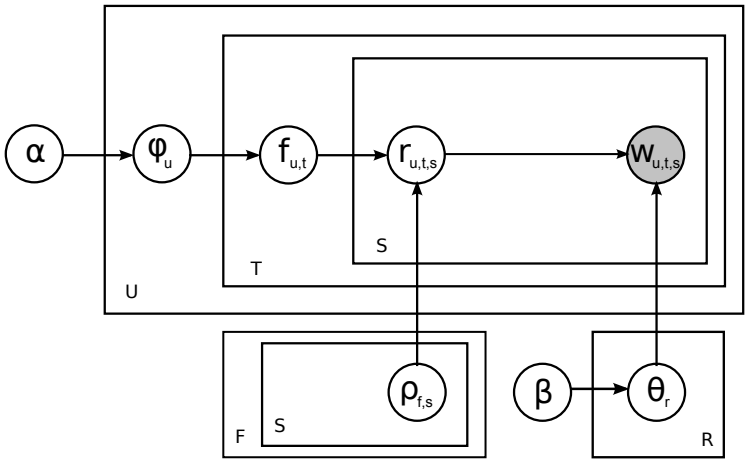


Fig. 1. Graphical model for LDA-Frames.

For the inference we use collapsed Gibbs sampling, where the θ, ρ and φ distributions are marginalized out. After having all topic variables **f** and **r** inferred, we can proceed to computing the lexical unit–frame distribution and the semantic role–word distribution. Let C^{φ}_{uf} be the count of cases where frame f is assigned to lexical unit u, C^{θ}_{rw} is the count of cases where word w is assigned to semantic role r and V is the size of vocabulary. The φ and θ distributions are computed using the following formulas:

$$\varphi_u = \frac{C^{\varphi}_{uf} + \alpha}{\sum_f C^{\varphi}_{uf} + F\alpha} \tag{1}$$

$$\theta_r = \frac{C^{\theta}_{rw} + \beta}{\sum_w C^{\theta}_{rw} + V\beta} \tag{2}$$

3 Measuring Semantic Relatedness

The semantic frames generated by the LDA-Frames algorithm are an interesting source of information about selectional preferences, but they can even be used for grouping semantically related lexical units. Separated semantic frames can hardly capture the whole semantic information about a lexical unit. Nevertheless, the LDA-Frames provide an information about the relatedness to every semantic frame we have inferred. After the inference process, each lexical unit u is connected with a probability distribution over semantic frames φ_u. Thus, we can group lexical units with similar probability distributions together to make a semantic cluster. In this work we will call these clusters *similarity sets*.

There are several methods how to compare probability distributions. We use the Hellinger Distance, which measures the divergence of two probability distributions, and is a symmetric modification of the Kullback-Leibner divergence [5]. For two probability distributions φ_a, φ_b, where $P(f|x)$ is the probability of frame f being generated by lexical unit x, the Hellinger Distance is defined as follows:

$$H(a,b) = \sqrt{\frac{1}{2}\sum_{f=1}^{F}\left(\sqrt{P(f|a)} - \sqrt{P(f|b)}\right)^2} \tag{3}$$

By using the Hellinger distance, we can generate a ranked list of semantically similar words for every lexical unit u the semantic frames have been computed for. Then the similarity set is chosen by selecting n best candidates or by selecting all candidates c, where $H(u,c) < \tau$ for some threshold τ.

4 The Experiment

The experiments have been performed for all transitive verbs having their lemma frequency in British National Corpus grater than 100. The transitiveness has been determined by selecting those verbs that have both subject valency and direct object valency presented in the Corpus Pattern Analysis lexicon [4]. Such constraints have been fulfilled by 4053 English verbs.

In order to generate LDA-frames for those English verbs, we have syntactically annotated British National Corpus using the Stanford Parser [2]. It has

provided a set of approximately 1.4 millions of (verb, subject, object) tuples that have been used as the training data for the LDA-frames algorithm. Based on previous experiments [8], we set the number of frames to 1200, number of roles to 400, and the hyperparameters as follows $\alpha = 0.1$, $\beta = 0.1$. After inferring φ distributions for every lexical unit, we have created a list of similar verbs sorted in ascending ordered based on the distance measure described in the previous section. The verbs with distance 1.0 were omitted. Using those data we have created 20 different thesauri. For $1 \leq n \leq 20$ the thesaurus has been built as the set of at most first n items from the similarity lists for every verb.

We have evaluated the quality of the thesauri built using LDA-frames by comparing them to the thesauri obtained from the Sketch Engine. To be fair, the word sketches have been generated on the British National Corpus just using the `subject_of` and `object_of` grammatical relations. The resulting thesaurus is in the form of sorted list of similar words, so we have been able to use the same method as in the case of the LDA-frames thesaurus and to create 20 thesauri in the same way.

It is obvious that not all verbs have got exactly n similar verbs in their similarity sets, because verbs with distance 1.0 were omitted. Table 1 shows average number of words in the similarity sets for every n we considered.

Table 1. Average number of words in the similarity sets.

n	1	2	3	4	5	6	7	8	9	10
LDA-frames	1.0	1.99	2.99	3.99	4.98	5.98	6.97	7.96	8.95	9.93
Sketch Engine	1.0	1.98	2.97	3.94	4.90	5.86	6.80	7.74	8.67	9.59
n	11	12	13	14	15	16	17	18	19	20
LDA-frames	10.91	11.89	12.87	13.85	14.82	15.79	16.76	17.73	18.69	19.66
Sketch Engine	10.50	11.40	12.30	13.18	14.05	14.92	15.77	16.61	17.45	18.27

The results from the table can be interpreted in the way that the Sketch Engine thesauri are stricter than LDA-frames ones and produce smaller similarity sets.

In order to measure the quality of the generated thesauri we have compared the similarity sets with synsets from English WordNet 3.0. First, we have selected verbs contained in both WordNet and our set of verbs. There were 2880 verbs in the intersection. The quality has been measured as the average number of verbs from a similarity set contained in the corresponding WordNet synset, normalized by the size of the similarity set. Formally, let V be the number of investigated verbs v_i, $T(v)$ similarity set for verb v in thesaurus T and $W(v)$ synset for verb v in WordNet:

$$Score(T) = \frac{1}{V} \sum_{v=1}^{V} \frac{|T(v) \cap W(v)|}{|T(v)|} \tag{4}$$

Resulting scores of both the Sketch Engine thesaurus and the LDA-frames thesaurus for similarity set sizes $1 \leq n \leq 20$ is shown in figure 2. One can see that the LDA-frames thesaurus outperforms the Sketch Engine for any choice of the size of similarity sets. The difference is most noticeable when $n = 1$. This special case measure whether the most similar verb is presented in the corresponding WordNet synset. This condition is satisfied in approximately 9.5 % verbs for LDA-frames and 6.5 % for Sketch Engine. The scores may seem to be quite small but it is important that only subject and object grammatical relations have been taken into consideration when computing the similarity. This means, for instance, that English verbs *eat* and *cook* have very high similarity scores, because they both are used in the same contexts and have completely identical semantic frames. It is straightforward that the algorithm could achieve much better results if there were used more than two grammatical relations. Specifically, verbs *eat* and *cook* could be differentiated, for example, by adding a grammatical relation corresponding with what instrument is used for eating or cooking.

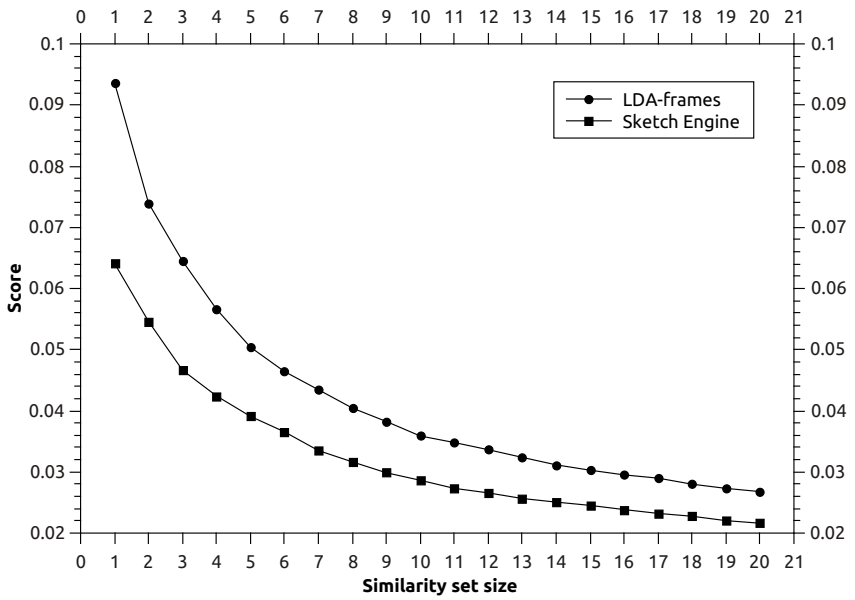


Fig. 2. Comparison of Sketch Engine thesaurus and LDA-frames thesaurus.

5 Conclusion

In this work we presented a new method for automatic building thesaurus from text corpora. The algorithm was applied to texts from the British National Corpus, and the quality was judged by measuring overlap with manually created synsets from WordNet 3.0. The results show that our algorithm outperforms similar approach from the Sketch Engine on the same training data. Only subject and object grammatical relation have been taken into consideration. In the future, we would like to enhance training data by other grammatical relation, which should lead to significantly better results.

Acknowledgements This work has been partly supported by the Ministry of Education of CR within the LINDAT-Clarin project LM2010013 and by EC FP7 project ICT-248307.

References

1. David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *J. Mach. Learn. Res*, 3:993 – 1022.
2. Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. Generating Typed Dependency Parses from Phrase Structure Parses. In *The International Conference on Language Resources and Evaluation (LREC) 2006*, 2006.
3. Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. The MIT Press, 1998.
4. Patrick Hanks and James Pustejovsky. A Pattern Dictionary for Natural Language Processing. In *Revue Francaise de Langue Appliquée*. Brandeis University, 2005.
5. Michiel Hazewinkel. *Encyclopedia of Mathematics*. Springer, 2001.
6. Werner Hüllen. *A History of Roget's Thesaurus: Origins, Development, and Design*. OUP Oxford, 2003.
7. Adam Kilgarriff, Pavel Rychlý, Pavel Smrž, and David Tugwell. The Sketch Engine. In *Proceedings of the Eleventh EURALEX International Congress*, pages 205–116. Lorient, France, 2004.
8. Jiří Materna. LDA-Frames: An Unsupervised Approach to Generating Semantic Frames. In Alexander Gelbukh, editor, *Proceedings of the 13th International Conference CICLing 2012, Part I*, pages 376–387, New Delhi, India, 2012. Springer Berlin / Heidelberg.

Improving Automatic Ontology Developement

Marek Grác, Adam Rambousek

Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
xgrac@fi.muni.cz, xrambous@fi.muni.cz

Abstract. This article describes the approach to build a new semantic network, which contains not only positive semantic labeling, but also the negative information. In order to obtain high quality data for the following use in machine learning and machine translation, we have created method based on automatically pre-generated data from the large corpora, followed by manual annotation. In this way, the core of semantic network was produced, which can be expanded to improve corpora coverage.

Key words: semantic network, ontology, ontology development

1 Introduction

To improve current complex NLP application we need to explore semantic level of natural languages. Both prevalent techniques (rule based and machine learning) tend to use information obtained from morphology and syntax level what lead to a situation where we cannot expect major improvement without adding something new. It can be completely new algorithms, more computing power or new data resources. In this paper we would like to focus on creating a new machine readable semantic network that will fill an existing gap.

Semantic networks and ontologies are used in NLP for several last decades. These networks focus on various aspects of semantic layer. There are ones which cover word senses (e.g. WordNet [1]) and other that are on the boundary of semantic and pragmatic layer of language (e.g. CyC [2]). The focus on semantic network is heavilly dependent on target applications. Ours goals are improving of syntactic parsing and as a result improve information extraction process from free text. Words in a language are very often bound to a specific set of words in language. These relations are traditionally called valencies.

Valencies, in various forms, are present for almost every PoS in language. The most studied ones are traditionally verb valencies. We have several verb valencies lexicons based on different linguistic theories which targets different usage. The most known are VerbNet [3], FrameNet [4] and Pattern Dictionary of English Verbs [5]. It is very important that such resources are consistent, they have acceptable coverage and that they are in superior quality. Such high expectations means that their development is expensive and unobtainable for smaller languages. Automatic methods of creating verb valencies using

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2012, pp. 105–110, 2012.

unsupervised methods of machine learning on unlabelled corpora are also available. Their quality vary a lot depending on language and used resources but in general hand-made (or semi-automatic) methods are still better even if they have lower coverage on text in corpora.

If we can automatically obtain valencies then we can create a thesaurus or even semantic network directly. But we can do it also in reverse order. If we have semantically labeled corpora obtaining a valency lexicon is no longer such difficult problem. In this paper we will show that we can iterativelly detect valencies and improve semantic network. In current state we cannot create valencies for verbs but we will use methods which are simple but suprisingly precise.

2 Semantic Network

Existing shallow semantic networks are influenced by Princenton's WordNet [1]. In fact WordNet is probably only network which lead to development of similar projects, for example different languages like EuroWordNet [6] and BalkaNet [7], or extensions like OntoWordNet [8] and eXtended WordNet [9].

For our experiment we have decided to use a Czech language because there is a Czech wordnet and thus ours results can be easilly compared to existing resources. As we expect to use proposed methods for languages without deep NLP support we will use only limited amount of technologies. Czech language is highly inflective with (almost) free word order in sentence. This means that direct usage of statistical methods will not work perfectly due to sparsness of data. For this purpose we will use morphological analyzer [10], guesser for unknown words and tagger / lemmatizer to obtain corpora with desambiguated morphological information. Used tagger 'desamb' [11] has precision sligtly over 90% and it's precision is very close to precision of taggers for other smaller languages. Existing Czech WordNet has almost same structure as Princeton one and we will use english names of elements across this paper.

When we take a look at verb valency dictionaries then the semantic class which is one of the most common is 'person' which is usually in position of 'agens'. In Czech language position of subject is in specific case 'nominative' but word-form representing this case is systematically same with case 'accusative' which represents object. Due to free word order, we are unable to obtain very high precision in this situation with just morpho-syntactic information.

Position of 'subject' with semantic class 'person' is very common but only very rarely subject is bound only to this specific class. More often there are also other semantic classes: institution and animals. These three classes are used in very similar way and it is very difficult to distinguish person and institution.

John loves Apple.
John loves Mary.

Simplest solution is to to create a new semantic class which will contain all such classes. Then we are in situation when John, Apple or bakery are in a same class because these words can be represented by person (or animal).

Bakery should also be in semantic class 'location' or 'building'. In WordNet-like networks this is done by adding new 'sense' of word which is hyponym for 'building'. We prefer not to fragment senses of words into separate categories and we do not target to an application that can uses such information. This was a main reason why we have decided to just add attributes to words. These attributes can be in hyponymy/hyperonymy relations.

We also believe that static (hard-coded) language resources are incomplete and they cannot be completed for living languge. This lead us to prefer open world assumptions (OWA) [12] in our ontology. OWA means that information which cannot be disapproved can be valid. Missing words in semantic network can have any available attributes. Because of OWA we have decided to populate the semantic network not only with positive information (John can have attribute *freewill*) but also negative information (table cannot have attribute *freewill*). Using such negative information helps us to properly work with words and valencies when we do not know what it is but we know that this can't be 'person' for example. In fact our preliminary results show us that these negative information are more usefull for syntactic parsing than positive ones. Mainly because quality of syntactic parser is so high that more often we will just confirm correct answer by semantic network but negative information will show us possible errors.

Problem with attributes instead of direct usage of hyponymy/hyperonymy relations is that we (in ideal world) have to work with every word in language. Expenses for annotation are then quite linear. For N semantic classes we have to answer N question for every word. Hypo/hypero relation between attributes can helps us to have N sufficiently low.

Annotation framework SySel [13] can be used to distribute and work with dividing words into categories. For yes/no questions the average answer is gathered from annotator in 1-2 seconds what lead to approx. 2,000 words / hour. Even if we need to annotate each word by several annotators, this process is really fast for smaller groups of words / semantic classes. If we need to distinguish attributes like can-it-looks-like-pumpkin? then it is very effecient way. Even if we have to handle usually only tens of thousands words (in our experiment approx. 100,000) then we would like to improve possibility to automatically add new words into semantic classes.

3 Extending existing semantic network

At the start of our experiment we did not focus on automatic methods of extending semantic network. Our decision was to create a completely new and free semantic network which will improve our existing tools. Creating a semantic network was not a goal of this process and that is main reason why our network has still huge gaps in semantic classes. We prefer do create a new semantic classes in a momement when we expect that they will improve tools not because we wish that they will maybe help one day.

In the first stage of project, 30 student annotators works with SySel to annotate 100,000 words if they can be *freewill* or they can't be. Each word was annotated at least two times and to accept a word/attribution/boolean, the metrics showed in table 1 were used.

Table 1. Accept metrics

# of annotators	non-agreement-annotation
2 - 4	0
5 - 8	1
9 - 12	2
12 +	3

We were able to create a semantic network that consists of 4,538 words which have attribute *freewill* and 101,303 that cannot have attribute *freewill*. More than 9,000 words didn't have the annotator agreement high enough to add them to semantic network. Relatively high level of inter-annotator error was probably due the fact of using work-power where some students did not focus on work (e.g. man is not *freewill* even if word is in examples) and only partially due to borderline words (e.g. democracy) that were not specified in annotation manual.

Semantic network have to be incomplete but we can attempt to improve it and extend it. We decided to test the most simple options of using existing semantic network and parsed corpora to do it.

In most of the language with free word order it is very difficult to match subject/verb and similar relations and full syntactic analysis is needed. But usually there are at least some rules that works pretty well. For Czech language we were able to identify those three:

- preposition, noun
- adjective, noun (if they have agreement in case, gender, number)
- noun, noun in genitive case - construction similar to english 'X of Y'

Very fast we found out that there is no preposition which is bound exclusively with *freewill*. Number of adjectives and nouns lead us to develop an automatic finder for such situations.

We want to find such words that are bound (almost) exclusively with given semantic class *freewill* using existing semantic network. From parsed corpora we will extract all bigrams which match our morphological rules. Then we will prepare stastistic for usage of each adjective with semantic class. These statistics is later filtered to contain only those adjectives which are used (almost) exclusively with words with possitive attribute *freewill*. Words which misses that attribute in semantic network (or they are not in network at all) will be accepted if there are enough adjectives that are used together with this word.

What are the main problems of this process?

- Our attributes means that word represents this attribute in one of its senses, this is important because in some cases we can detect adjective together with word-sense which we do not plan. e.g five-pointed star vs rock star.
- We can find out adjectives which are only partially relevant. Quite a lot of found adjective belongs to group of adjective that represents 'X years old'. Our system correctly does not find one-year old, five-years old because there are lot of mentions of wine, whiskey, buildings, ... and it will correctly find 72-years old as such numbers are usually specific for person. Very similar process is in place for possesive adjective (e.g. dog's, Achilles's).
- As we are working with bigrams directly it is possible that we will add a word which do not have semantic representation of attribute directly. e.g He ate a "box" of chocolate. The word 'box' works just as a modifier in this context. We can detect these words because they will occur as positive examples for most of the attributes.

Process itself is relatively fast and on 350 million corpora it took less than 3 hours to make one iteration. Due to quality of data we can add found words to semantic network automatically and re-run the process. Our experiments showed that few iteration will drastically improve coverage and this method can very easilly solve border-line cases where human annotators are not sure. Border-line cases does not have to be solved consistently but we can add words only to positive side of semantic network.

tabulka s vysledkami

Table 2. Coverage

	# of words identified	k1 coverage	'k2 k1' coverage
manual	105,840	68.26%	94.28%
after 1st iteration	106,044	74.48%	96.08%
after 2nd iteration + selected proper nouns[1]	106,250	81.49%	97.99%
after 3rd iteration	106,942	83.07%	98.6%

Table 3. Random samples of 10,000 words with *freewill* attribute, called seed1, seed2, seed3

sample	k1 coverage	new words precision
seed1, iteration 2	25.51%	84.50% (780 words)
seed2, iteration 2	40.76%	75.78% (1514 words)
seed2, iteration 3	33.19%	72.24% (788 words)

4 Evaluation and future extensions

As seen in table 3, automatic detection algorithm combining manual annotation of small starting set, valency detection, and limiting linguistic rules works very well in identifying word attributes, ie. can say that word belongs to particular semantic class.

Negative attributes, ie. information that word does not belong to semantic class, are very useful feature for the applications using the semantic network. Such knowledge can reduce the time needed to parse the text, for example. However, negative attributes in our semantic network are annotated manually only. Current rules and tools for automatic annotation does not provide precision good enough to include in semantic network. Improvement of negative attributes detection is one of the next steps of this project. Coverage enhancement is the other big goal, both in terms of annotated words, and different semantic classes.

Acknowledgements This work has been partially supported by the Ministry of Education of CR within the LINDAT-Clarin project LM2010013 and by EC FP7 project ICT-248307.

References

1. Fellbaum, C., ed.: WordNet: An Electronic Lexical Database. MIT Press (1998)
2. Lenat, D., Guha, R., Pittman, K., Pratt, D., Shepherd, M.: Cyc: toward programs with common sense. Communications of the ACM **33**(8) (1990) 30–49
3. Schuler, K.: VerbNet: A broad-coverage, comprehensive verb lexicon. PhD thesis, University of Pennsylvania (2005)
4. Fillmore, C., Baker, C., Sato, H.: Framenet as a 'net'. In: Proceedings of Language Resources and Evaluation Conference (LREC 04). Volume vol. 4, 1091-1094., Lisbon, ELRA (2004)
5. Hanks, P., Pustejovsky, J.: A pattern dictionary for natural language processing. Revue Française de linguistique appliquée **10**(2) (2005) 63–82
6. Vossen, P., ed.: EuroWordNet: a multilingual database with lexical semantic networks for European Languages. Kluwer (1998)
7. Christodoulakis, D.: Balkanet Final Report, University of Patras, DBLAB (2004) No. IST-2000-29388.
8. Gangemi, A., Navigli, R., Velardi, P.: The ontowordnet project: extension and axiomatization of conceptual relations in wordnet. On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE (2003) 820–838
9. Mihalcea, R., Moldovan, D.: extended wordnet: Progress report. In: Proceedings of NAACL Workshop on WordNet and Other Lexical Resources. (2001) 95–100
10. Šmerk, P.: Fast morphological analysis of czech. RASLAN 2009 Recent Advances in Slavonic Natural Language Processing (2009) 13
11. Šmerk, P.: K morfologické desambiguaci češtiny. (2008)
12. Reiter, R.: On closed world data bases. Technical report, Vancouver, BC, Canada, Canada (1977)
13. Grác, M., Rambousek, A.: Low-cost ontology development. (In: 6th International Global Wordnet Conference Proceedings) 299–304

Authorship Verification based on Syntax Features

Jan Rygl, Kristýna Zemková, Vojtěch Kovář

NLP Centre
Faculty of Informatics
Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
xrygl@fi.muni.cz

Abstract. Authorship verification is wildly discussed topic at these days. In the authorship verification problem, we are given examples of the writing of an author and are asked to determine if given texts were or were not written by this author. In this paper we present an algorithm using syntactic analysis system SET for verifying authorship of the documents. We propose three variants of two-class machine learning approach to authorship verification. Syntactic features are used as attributes in suggested algorithms and their performance is compared to established word-lenth distribution features. Results indicate that syntactic features provide enough information to improve accuracy of authorship verification algorithms.

Key words: authorship verification, syntactic analysis

1 Introduction

The interest in autorship verification can be found in 18th century in the Shakespearean era. A lot of linguists wanted to prove (or disprove) that William Shakespeare wrote the well known plays [1]. After that, this topic was discussed more and more often.

The task of authorship verification is commonly distinguished from that of authorship attribution. In both text classification approaches, the task is to decide whether a given text has been written by a candidate author. In authorship attribution, the actual author is known to be included in the set of candidates (closed case). In authorship verification, however, this assumption cannot be made: the given text might have been written by one of the candidate authors, but could also be written by none of them (open case). Note that this scenario is typical of forensic applications where it cannot be presupposed that the author of, for example, a letter bomb is among the suspect candidate authors.[2]

There are three main approaches to the authorship verification:

1. One-Class Machine Learning: In this approach[3] authors used only positive examples for training, because they consider difficult to select representative negative examples.

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2012, pp. 111–119, 2012.

2. Two-Class Machine Learning: The technique is established for an authorship attribution task.
3. Unmasking algorithm: The main assumption is that only small number of features distinguish between authors. The most distinguishing features are iteratively removed. Hypothesis is that whatever differences there are between document will be reflected in only a relatively small number of features. [4]

So far, not many authors have specialized their work to authorship verification, especially with syntactic features. The availability of fast and accurate natural language parsers allow for serious research into syntactic stylometry. [5]

In this paper, we focus on syntactic features combined with the Two-Class Machine Learning approach. The unmasking algorithm requires several different types of features and One-Class Machine Learning performs worse than Two-Class ML [3]. Three implementations of Two-Class ML approach are tested. The basic variant using Support Vector Machines is utilized and two modifications are suggested:

– The authorship verification problem is converted to the authorship attribution task by adding several random documents.
– In the authorship attribution problem, similarities of documents are transformed to rankings of documents. [6]

Because, unlike other publications, we work with texts consisting of up to tens of sentences, we have to cope with insufficiency of qualitative and quantitative information. Not many linguistics have focused on short texts because not enough material can cause lower accuracy.

2 Syntactic Analysis of the Czech Language

The main aim of the natural language syntactic analysis is to show the surface structure of the input sentence. Czech is one of the free-word-order languages with rich morphology that poses barriers to parsing using formalisms that are relatively succesfull when used with fixed-word-order languages such as English. Because of unrestricted word order in Czech, current Czech parsers face problems such as high ambiguity of the analysis output or low precision or coverage on corpus texts.

There are three main approaches to the automatic syntactic analysis of Czech at this time. The first uses the formalism of Functional Generative Description, FGD, for syntax description and is developed at the Institute of Formal and Applied Linguistics in Prague. Within this formalism, the syntactic information is encoded as an acyclic connected graph of dependency relations, called *dependency tree*. [7]

The second approach to Czech language parsing, `synt`, uses the constituent formalism of syntax and its development centre is located at the Natural Language Processing Centre of Masaryk University in Brno. The constituent

formalism encodes the syntactic information as a derivation tree based on the formal grammar of the language. System `synt` is based on a metagrammar formalism with a context-free backbone, contextual actions and an efficient variant of the chart parsing algorithm. [8]

The third approach is system SET. This open source tool was developed at the NLP Centre as well. SET is based on the simple principle of pattern matching, so it is fast, understandable for people and easily extensible. It is written in Python which means it is easily usable on different platforms and there is no need for complicated installation. The core of SET consists of a set of patterns (or rules) and a pattern matching engine that analyses the input sentence according to given rules. Currently, SET is distributed with a set of rules for parsing the Czech language, containing about 100 rules. The primary output of the analysis is a *hybrid tree* – a combination of constituent and dependency formalism – but SET also offers converting this tree into purely dependency or purely constituent formalism. Other output options include extraction of phrases in several settings, finding dependencies among these phrases or extraction of collocations.

3 Extracting Syntax Features using SET

Nowadays, SET is one of the fastest available parsing systems for Czech with reasonable precision, it is freely available and very easy to use. Therefore we decided to use it for extraction of syntactic features in our experiment. As outlined above, SET produces parsing trees in three possible output fomats: dependency format (`-d` option), constituent format (`-p` option) and hybrid format (default). Dependency and constituent tree is illustrated in Figure 1, for Czech sentence *Verifikujeme autorství se syntaktickou analýzou.* (*We verify the authorship using syntactic analysis.*), as analyzed by SET. On the left hand side, we can see a phrasal tree; on the right side, a dependency tree.

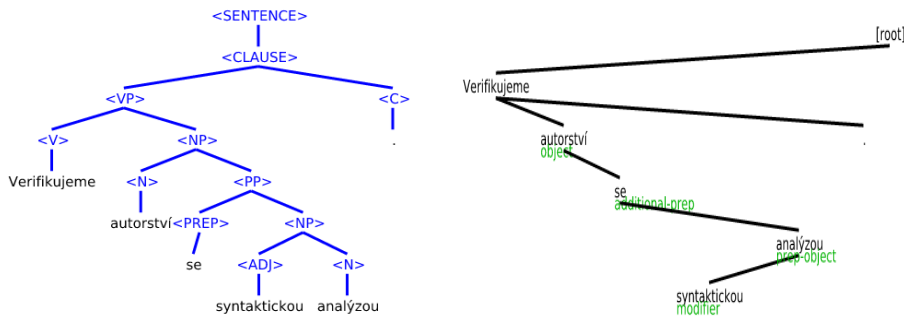


Fig. 1. Dependency and phrasal tree from SET

The dependency and phrasal output of SET was used to extract features for machine learning of differences among the authors. Namely, the following features were used:

- maximum depth of the dependency tree
- highest number of child nodes in the dependency tree
- absolute and relative frequencies of particular non-terminals in the phrasal tree (e.g. <*CLAUSE*>, <*NP*>, <*VP*>)
- absolute and relative frequencies of particular dependency labels in the dependency tree (e.g. *prep-object*, *verb-object*)

4 Authorship Verification Algorithms

In authorship verification problem, we are given two documents A and B and are asked to determine if documents were or were not written by the same author.

Two-Class Machine Learning algorithm was implemented and other two algorithms were designed to verify that two documents were written by the same author.

1. **Two-Class Machine Learning:**
   Basic approach to Authorship Verification is to train Machine Learning model to decide if two documents A, B do or do not have the same author. The main disadvantage is that it is impossible to cover all types of negative examples in training data.

   given $document_A$, $document_B$, empty *attributeList*
   for *i* in *1 ...count(features)*:
   feature = *features[i]*
   $attributeList[i] = |feature(document_A) - feature(document_B)|$
   Model(attributeList) predicts if documents were written by same author.

2. **Converting verification to attribution problem:**
   "Authorship verification ... generally deemed more difficult than so-called authorship attribution."[2], therefore we transformed problem by adding 4 documents $D_1, \ldots, D_4$. Attribution method selects from candidates B, D_1, $\ldots$, D_4 the most similar document to A. If the document B is selected with enough confidence, documents A and B are written by same author.

   given $document_A$, $document_B$, empty *attributeList*
   select 4 random documents (D_1, D_2, D_3, D_4) of similar length to $document_B$
   for *doc* in ($document_B$, D_1, D_2, D_3, D_4):
   empty *attributeList*
   for *i* in *1 ...count(features)*:
   feature = *features[i]*
   $attributeList[i] = |feature(document_A) - feature(doc)|$
   Model(attributeList) computes probability $prob_{doc}$ of same authorship

```
if prob_B >= 0.5 ∧ prob_B = max(prob_B,1,2,3,4): "same authorship"
```

3. **Algorithm 2 extended by replacing similarity scores by their rankings:** Our previous experiments showed that accuracy of Authorship Attribution problem can be improved by replacing similarities of documents by their rankings. [6]

```
given document_A, document_B, empty attributeList
select 4 random documents (D_1, D_2, D_3, D_4) of similar length to document_B
for i in 1 ...count(features):
        feature = features[i]
        rank = 1
        diff = |feature(document_A) − feature(document_B)|
        for doc in (D_1, D_2, D_3, D_4):
              if |feature(document_A) − feature(doc)| < diff: rank + =
1
        attributeList[i] = 1/rank
Model(attributeList) predicts if documents were written by same author.
```

5 Experiments

Data

400 Czech documents (10 documents per author) downloaded from the Internet were used. The data were collected from Czech blogs and Internet discussions connected to these blogs and were preprocessed automatically by the Czech morphological tagger *Desamb* [9] and the *SET* parser [7]. The document length ranges from 1 to about 100 sentences.

Machine Learning

LIBSVM [10] implementation of Support Vector Machines algorithm was selected as the machine learning component and 4-fold cross-validation was used for evaluation.

Authors were divided into 4 groups, each group contained 10 authors and 100 documents. During all experiments, authors of learning documents were different to authors of test documents.

Models were trained utilizing 1000 positive and 1000 negative examples for each scenario. To create positive examples, documents A and B were randomly selected from the same author; to simulate negative examples, an author of document B was different to the author of A. Authors of documents $D_1, \ldots, D_4$ used in algorithm 2 and 3 were different to authors of A and B for both positive and negative examples.

Algorithm 1: Two-Class ML

For the basic algorithm, the average accuracy was **57.9 %** (7.9 % over the baseline). Detailed results are shown in Table 1.

Table 1. Results of Algorithm 1

(a) Folder 1: Accuracy: 51.1 %

	Positive	Negative
True	280 (38.5 %)	92 (12.6 %)
False	272 (37.4 %)	84 (11.5 %)

(b) Folder 2: Accuracy: 55.4 %

	Positive	Negative
True	360 (41.7 %)	119 (13.8 %)
False	313 (36.2 %)	72 (8.3 %)

(c) Folder 3: Accuracy: 67.7 %

	Positive	Negative
True	230 (33.6 %)	233 (34.1 %)
False	109 (15.9 %)	112 (16.4 %)

(d) Folder 4: Accuracy: 57.2 %

	Positive	Negative
True	224 (28.7 %)	222 (28.5 %)
False	168 (21.5 %)	166 (21.3 %)

```
Folder 1: Train accuracy 77.4 % for parameters c=2.0 g=0.5
Folder 2: Train accuracy 75.5 % for parameters c=8.0 g=0.5
Folder 3: Train accuracy 70.2 % for parameters c=2048.0 g=0.125
Folder 4: Train accuracy 73.3 % for parameters c=2048.0 g=0.125
```

Algorithm 2: Converting Authorship Verification to Attribution

This method was found to be unsuitable to solve Authorship Verification problem. Average accuracy did not even exceed the baseline.

Algorithm 3: Converting Authorship Verification to Attribution using Ranking instead of Score

With the last algorithm, the average accuracy was **71.3 %**. If we consider short lengths of documents, obtained results are good. Accuracy of this method is 21.3 % better than the baseline and represents 13.5 % improvement over algorithm 1). See detailed results in Table 2.

Performance Comparison: Word-Length Distribution

Word-Length approach published by T. C. Mendenhall in 1887 [11] is still used in many current works. To compare our syntactic features with this approach, we replaced them by the word-length distribution and then used the same algorithms.

– **Algorithm 1:** Two-Class ML with Word-Length Features
Average accuracy is **53.2 %**, which is only slightly better than the baseline. Detailed results are shown in Table 3.

Table 2. Results of Algorithm 3

(a) Folder 1: Accuracy: 79.3 %

	Positive	Negative
True	691 (34.6 %)	894 (44.7 %)
False	106 (5.3 %)	309 (15.4 %)

(b) Folder 2: Accuracy: 64.3 %

	Positive	Negative
True	364 (18.2 %)	921 (46.0 %)
False	79 (4.0 %)	636 (31.8 %)

(c) Folder 3: Accuracy: 69.0 %

	Positive	Negative
True	481 (24.1 %)	899 (44.9 %)
False	101 (5.1 %)	519 (25.9 %)

(d) Folder 4: Accuracy: 72.8 %

	Positive	Negative
True	491 (24.6 %)	965 (48.2 %)
False	35 (1.8 %)	509 (25.4 %)

```
Folder 1: Train accuracy 88.9 % for parameters c=512.0 g=0.125
Folder 2: Train accuracy 88.2 % for parameters c=2048.0 g=2.0
Folder 3: Train accuracy 88.0 % for parameters c=8.0 g=2.0
Folder 4: Train accuracy 87.7 % for parameters c=8.0 g=2.0
```

Table 3. Results of Algorithm 1

(a) Folder 1: Accuracy: 52.9 %

	Positive	Negative
True	404 (44.9 %)	72 (8.0 %)
False	378 (42.0 %)	46 (5.1 %)

(b) Folder 2: Accuracy: 53.0 %

	Positive	Negative
True	358 (39.8 %)	119 (13.2 %)
False	331 (36.8 %)	92 (10.2 %)

(c) Folder 3: Accuracy: 50.1 %

	Positive	Negative
True	326 (36.2 %)	125 (13.9 %)
False	325 (36.1 %)	124 (13.8 %)

(d) Folder 4: Accuracy: 56.9 %

	Positive	Negative
True	358 (39.8 %)	154 (17.1 %)
False	296 (32.9 %)	92 (10.2 %)

```
Folder 1: Train accuracy 77.8 % for parameters c=8.0 g=0.125
Folder 2: Train accuracy 77.9 % for parameters c=2.0 g=0.5
Folder 3: Train accuracy 80.0 % for parameters c=2.0 g=0.5
Folder 4: Train accuracy 79.4 % for parameters c=8192.0 g=0.0078125
```

– **Algorithm 3:** Authorship Attribution with Rankings replacing Scores (with Word-Length Features)

  Average accuracy is **61.5 %**. The train accuracies indicate that the machine learning model is partially overfitted. The accuracy could be slightly increased by further optimizations, involving heuristic selection of attributes, but given described size of the learning set and lengths of documents, word-length features are outperformed by our syntactic features. Results are displayed in Table 4.

Table 4. Results of Algorithm 3

(a) Folder 1: Accuracy: 62.7 %

	Positive	Negative
True	259 (12.9 %)	994 (49.7 %)
False	6 (0.3 %)	741 (37.1 %)

(b) Folder 2: Accuracy: 61.4 %

	Positive	Negative
True	229 (11.4 %)	998 (49.9 %)
False	2 (0.1 %)	771 (38.6 %)

(c) Folder 3: Accuracy: 62.2 %

	Positive	Negative
True	244 (12.2 %	999 (49.9 %)
False	1 (0.1 %)	756 (37.8 %)

(d) Folder 4: Accuracy: 59.8 %

	Positive	Negative
True	196 (9.8 %)	1000 (50.0 %)
False	0 (0.0 %)	804 (40.2 %)

```
Folder 1: Train accuracy 91.9 % for parameters c=2.0 g=2.0
Folder 2: Train accuracy 91.3 % for parameters c=2.0 g=2.0
Folder 3: Train accuracy 91.1 % for parameters c=8.0 g=2.0
Folder 4: Train accuracy 90.7 % for parameters c=8.0 g=2.0
```

6 Conclusions and Future Work

The primary aim of this paper was to present a syntactic approach to the authorship verification task. Because, unlike other publications, we work with texts consisting of up to tens of sentences, we have to cope with insufficiency of qualitative and quantitative information. Despite the fact that the accuracy of our method does not achieve desired results yet, the experiment indicates that syntactic features can outperform established approaches.

Within the future work, our goal is to find another syntactic atributes to add to our algorithms. We also plan combining syntactical and morphological information together.

Acknowledgments

This work has been partly supported by the Ministry of the Interior of CR within the project VF20102014003.

References

1. Malone, Edmond: A Dissertation on the Three Parts of King Henry VI. Tending to Shew That Those Plays Were Not Written Originally by Shakspeare. Gale Ecco, Print Editions (1787)
2. Kestemont, M., Luyckx, K., Daelemans, W., Crombez, T.: Cross-genre authorship verification using unmasking. English Studies **93**(3) (2012) 340–356
3. Manevitz, L.M., Yousef, M., Cristianini, N., Shawe-taylor, J., Williamson, B.: One-class svms for document classification. Journal of Machine Learning Research **2** (2001) 139–154

4. Koppel, M., Schler, J.: Authorship verification as a one-class classification problem. In: Proceedings of the twenty-first international conference on Machine learning. ICML '04, New York, NY, USA, ACM (2004) 62–
5. Hollingsworth, C.: Using dependency-based annotations for authorship identification. In Sojka, P., Horák, A., Kopeček, I., Pala, K., eds.: Text, Speech and Dialogue. Volume 7499 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2012) 314–319
6. Rygl, J., Horák, A.: Similarity ranking as attribute for machine learning approach to authorship identification. In Chair), N.C.C., Choukri, K., Declerck, T., Doğan, M.U., Maegaard, B., Mariani, J., Odijk, J., Piperidis, S., eds.: Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12), Istanbul, Turkey, European Language Resources Association (ELRA) (2012)
7. Kovář, V., Horák, A., Jakubíček, M.: Syntactic analysis using finite patterns: A new parsing system for czech. In Vetulani, Z., ed.: LTC. Volume 6562 of Lecture Notes in Computer Science., Springer (2009) 161–171
8. Horák, A.: Computer Processing of Czech Syntax and Semantics. Librix.eu, Brno, Czech Republic (2008)
9. Šmerk, Pavel: K počítačové morfologické analýze češtiny. PhD thesis, Faculty of Informatics Masaryk University (2010)
10. Chang, Chih-Chung - Lin, Chih-Jen: LIBSVM: a library for support vector machines. (2001) URL: `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.
11. Mendenhall, T. C.: The characteristic curves of composition. The Popular Science **11** (1887) 237–246

Segmentation from 97% to 100%
Is It Time for Some Linguistics?

Petr Sojka

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

Abstract. Many tasks in natural language processing (NLP) require *segmentation* algorithms: segmentation of paragraph into sentences, segmentation of sentences into words is needed in languages like Chinese or Thai, segmentation of words into syllables (*hyphenation*) or into morphological parts (e.g. getting word stem for indexing), and many other tasks (e.g. tagging) could be formulated as segmentation problems. We evaluate methodology of using *competing patterns* for these tasks and decide on the complexity of creation of space-optimal (minimal) patterns that completely (100 %) implement the segmentation task. We formally define this task and prove that it is in the class of *non-polynomial* optimization problems. However, finding space-efficient competing patterns for real NLP tasks is feasible and gives efficient scalable solutions of segmentation task: segmentation is done in *constant* time with respect to the size of segmented dictionary. Constant time of access to segmentations makes competing patterns attractive data structure for many NLP tasks.

Key words: competing patterns, segmentation, hyphenation, NP problems, pattern generation, patgen, context-sensitive patterns, machine learning, natural language engineering

> Everything is a symbol, and symbols can be combined to form *patterns*. Patterns are beautiful and revelatory of larger truths. These are the central ideas in the thinking of Kurt Gödel, M. C. Escher, and Johann Sebastian Bach, perhaps the three greatest minds of the past quarter-millennium. (Hofstadter [1])

1 Introduction

Many tasks in NLP require *segmentation* algorithms: segmentation of paragraph into sentences, segmentation of sentences into words is needed in languages like Chinese or Thai, segmentation of words into syllables (*hyphenation*) or into morphological parts (e.g. getting word stem for indexing), phoneme (speech) segmentation, and many other tasks (e.g. tagging) could be expressed as segmentation problems. Solution of segmentation task is an important brick in every NLP framework.

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2012, pp. 121–131, 2012.

As the available computing power steadily grows, new approaches recently deemed impossible are becoming reality – *empirical approaches* are used for machine learning of language phenomena: from huge language data (corpora, wordlists), language models and patterns are learnt by sophisticated algorithms through *machine learning* techniques. As examples of this shift, successful unsupervised learning of natural language morphology from language word lists has been reported in [2], and as overviewed in [3], supervised learning approaches are very successful for various types of segmentation. These merely statistical approaches work quite well for many tasks in the area of computational linguistics, and quickly reach above 90% efficiency in tasks such as part of speech tagging, sentence segmentation, speech recognition or probabilistic parsing. The main drawback of a solely statistical approach is that the results of learning methods are usually not understandable by expert linguists, as the language models are hidden in weights of synapses of neural nets or in zillions of probabilities or conditioned grammar rules. It appears that going the "last mile", increasing the remaining few percent purely statistically is not feasible, and ways to cover the remaining exceptions by usage of symbolic, linguistic descriptions are being sought [4].

Recognition of patterns is considered as the central issue in intelligence. Artificial intelligence needs *statistical emergence* [5]: for real semantics, symbols must be *decomposable,* complex, autonomous – active. A rule-based approach, such as, when the results of the learning process are human-understandable *rules* or *patterns*, allows for the merging of hand-crafted and machine learnt knowledge. It is becoming clear that a close cooperation between computer scientists and linguists is necessary [6] – both sides need each other. Neither rigorous computational analysis and formal models nor linguistic introspection and language models should be absent in successful approaches. First, symbolical descriptions should be sought, and only when not sufficient a dice should be drawn.

Patterns can be identified as a set of objects that share some common property. During the emergence of patterns covering the rules in data, some *exceptions* may occur. Remaining errors and exceptions covered in the first level can be viewed again as set of objects and described by *inhibiting patterns*. The next layer of *covering patterns* may describe the patterns in the data not handled by previous rules, and so on. By this process, knowledge from the data can be learnt, either by an automatic procedure, or by information fusion from different sources.

There is plethora of methods of machine learning, data mining and knowledge management. However, up to now, we are not aware of an systematic attempt made to deal with the large-scale exception handling that is so widespread across linguistic data in machine learning methods and data mining. This work is one of the first attempts to formalize and fully employ the theory of competing patterns for the utilization of language data in the areas of natural language processing and computer typesetting.

2 Basic Notions

The two fundamental problems are pattern definition and pattern recognition/generation from input data. There are many ways of formalizing patterns – sets of objects sharing some recognizable properties (attributes, structure, ...).

Definition 1 (pattern). *By alphabet we mean a finite, nonempty set. Let us have two disjoint alphabets* Σ *(the alphabet of* terminals*, called also called* characters *or* literals*) and* V *(the alphabet of* variables*).* Patterns *are words over the free monoid* $\langle(\Sigma \cup V)^*, \varepsilon, \cdot\rangle$*. The length* $|\varepsilon|$ *of an empty word* ε *is zero. Patterns having only terminals are called* terminal patterns *or* literal patterns*. The length of a literal pattern* p*, denoted by* $|p|$*, is the number of literals in it. The* language $L(\alpha)$ defined by a pattern α *consists of all words obtained from* α *by leaving the terminals unchanged and substituting a terminal word for each variable* v*. The substitution in our case has to be* uniform*: different occurences of* v *are replaced by the same terminal word. If the substitution always replaces variables by a* nonempty *word, such language* L_{NE} *is* non-erasing*, and such pattern is called* NE-pattern*. Similarly, we define an* erasing *language* L_E *as a language generated by an* E-pattern *such that substitution of variable* v *by empty word* ε *is allowed.*

The pattern *SVOMPT* for English sentences where the variables denote Subject, Verb, Object, Mood, Place, Time may serve as an example of E-pattern. A useful task is to infer a pattern common to all input words in a given sample by the process of *inductive inference*. It has been shown by Jiang et al. [7] that the *inclusion problem* is undecidable for both erasing and non-erasing pattern languages. It is easy to show that the decidability of the *equivalence problem* for non-erasing languages is trivial. The decidability status of the equivalence problem for E-patterns remains open. These results show that trying to infer language description in the form of a set of patterns (or the whole grammar) automatically is very difficult task.

We focus our attention in the further study to literal patterns only.

Definition 2 (classifying pattern). *Let* A *be alphabet, let* $\langle A, \leq\rangle$ *be a partially ordered system,* $\leq$ *be a* lattice order *(every finite non-empty subset of* A *has lower and upper bound). Let* . *be a distinguished symbol in* $\Sigma' = \Sigma \cup \{.\}$ *that denotes the beginning and the end of word* – begin of word marker *and* end of word marker. Classifying patterns *are the words over* $\Sigma' \cup V \cup A$ *such that dot symbol is allowed only at the beginning or end of patterns.*

Terminal patterns are "context-free" and apply anywhere in the classified word. It is important to distinguish patterns applicable at the beginning and end of word by the dot symbol in a pattern.[1] Classifying patterns allow us to build *tagging hierarchies* on patterns.

[1] It is itmrnopt to dgtusisinh ptatren apcbliplae at the bngninieg and end of wrod by the dot sobmyl in a ptarten.

Definition 3 (word classification, competing word patterns).
*Let P be a set of patterns over $\Sigma' \cup V \cup A$ (*competing patterns, pattern set*). Let $w = w_1 w_2 \ldots w_n$ be a word to be classified with P. Classification classify$(w, P) = a_0 w_1 a_1 w_1 \ldots w_n a_n$ of w with respect to P is computed from a pattern set P by a competing procedure: all patterns whose projection to Σ match a substring of w are collected. a_i is supremum of all values between characters w_i and w_{i+1} in matched patterns. classify(w, P) is also called the* winning pattern.

It is worth noting that the classification procedure can be implemented very efficiently even for large pattern bases. Its effectiveness depends on the data structure where the patterns are stored. When an indexed trie is used, the classification of a word can be realized in linear time with respect to the word length $|w|$ and does not depend on $|P|$.

Our motivation for studying of competing patterns was the word division (hyphenation) problem. It is related to a dictionary problem – the problem of effective storage of a huge word list. An enumerated list of Czech words may have well above 6,000,000 words. Storage of such a large table even using hashing requires considerable space. Another idea is to use finite-state methods – finite-state automata and transducers. It has been shown that decomposition of the problem by using *local grammars* [8] or building cascades of finite state machines [9] is a tractable, even though very time-consuming task. The main problem with these approaches is that they do not generalize well – they do not perform well on unseen words. A structural decomposition of W into patterns is the key idea here, and brings better generalization qualities:

Definition 4 (word division problem). *Let W be a set of words over $\Sigma \cup \{0, 1\}$ such that placing 1 between two letters in w denotes the possibility of word division at that point (placing 0 or nothing means the opposite). We want to find pattern set P such that winning patterns classify(w, P) encode the same information as w. In this case we say that P or classify(w, P)* covers *w.*

We want to find a pattern set that is minimal in size and maximal in performance; we have to define these performance measures.

Definition 5 (precision, recall, F-score). *Let $W = (\Sigma \cup \{0, 1\})^*$, and P a set of patterns over $\Sigma' \cup \mathbb{N}$. Let good(w, P) is the number of word divisions where classify(w, P) covers w, good$(W, P) = \sum_{w \in W}$ good(w, P). bad(w, P) is the number of word divisions where classify(w, P) classifies word division that is not in w, bad$(W, P) = \sum_{w \in W}$ bad(w, P). missed(w, P) is the number of word divisions where classify(w, P) fails to classify word division that is in w, missed$(W, P) = \sum_{w \in W}$ missed(w, P). The definition of the measures is then as follows:*

$$\mathit{precision}(W, P) = \frac{\mathit{good}(W, P)}{\mathit{good}(W, P) + \mathit{bad}(W, P)} \tag{1}$$

$$\mathit{recall}(W, P) = \frac{\mathit{good}(W, P)}{\mathit{good}(W, P) + \mathit{missed}(W, P)} \tag{2}$$

The precision and recall scores can be combined into a single measure, known as the *F-score* [10]:

Definition 6 (F-score).

$$F(W,P) = \frac{2 \times precision(W,P) \times recall(W,P)}{precision(W,P) + recall(W,P)} \tag{3}$$

An F-score reaches its maximum when both precision and recall is maximal; in the case $F(W,P) = 1$ all information about word division is compressed into the pattern base P.

Definition 7 (lossless compression, cross validation). *If $F(W,P) = 1$ we say that we* losslessly compressed *W into P. We can test performance of P on an unseen word list W' to measure the generalization properties of pattern set P – in the machine learning community, the term* cross validation *is used.*

3 Generation of Minimal Patterns is Non-Polynomial Task

Here we see one advantage of the pattern approach. In the case where we have solved the hyphenation problem by storing all the words with the division point in a hash table or using a finite state transducer, we do not know how to segment new, unseen words. On the other hand, pattern P trained on W can perform well on unseen words (typically new long words or compounds) – as in patterns the *rules* are generalized.

There are many pattern sets P that losslessly compress (cover) W; one straightforward solution is having just one pattern for every word $w \in W$ by putting dot symbol around the word with division points marked by 1. Such a pattern set P is a *feasible solution*. But we want to obtain minimal pattern set. Minimality can be measured by the number of patterns, by the number of characters in patterns, or by the space the patterns occupy when stored in some data structure. Even if we take the simplest measure by counting the patterns, and try to find a minimal set of patterns that cover W, we will show how hard the task is. To formulate it more precisely, we need to define:

Definition 8 (minimum set cover problem). *An instance of set cover problem is finite set X and a family $\mathcal{F}$ of subsets of X, such that $X = \bigcup_{S \in \mathcal{F}} S$. The problem is to find a set $C \subseteq \mathcal{F}$ of* minimal size *which covers X, i.e. $X = \bigcup_{S \in C} S$.*

The minimum set cover problem (MSCP) is known to be in the class of NPO problems (optimization problems analogical to NP decision problems), [11]. A variant of MSCP, in which the subsets have positive weights and the objective is to minimize the sum of the weights in a set cover, is also NPO. Weighted version of minimum set cover problem is approximable within $1 + ln|X|$ as shown by Chvátal [12].

Theorem 1 (pattern minimization problems). *Let W be a set of words with one division only. Problem of finding minimal number of patterns P that losslessly compress W is equivalent to the (weighted) minimum set cover problem.*

Proof. We show that the problem reduces to the minimal set cover problem. For every subset $C \in W$ there exists at least one feasible solution P_C such that P_C covers C and does not cover any word in $W \setminus C$, e.g., pattern set $\{.c. \mid c \in C\}$. Between all such feasible solutions we choose a canonical representative P'_C – a set which is smallest by some measure (e.g., number of patterns, or number of characters in the pattern set). We now have a one to one correspondence between all pattern sets that cover exactly C represented by P'_C and C. Thus we showed that a pattern coverage minimization problem is equivalent to the weighted minimum set cover [12] in NPO class.

We have shown that even a pattern covering problem without competition is already NPO. When trying to cover W by competing patterns, complicated interactions may arise – we need some approximation of the optimal solution.

Liang's main concern in the pattern covering problem was the size of the patterns stored in a packed trie (indexed trie with packing the different families of the trie into a single large array in computer memory. He discusses NP-completeness of finding a minimum size trie [13, page 25] by pointing to the problem transformation from graph coloring by Pfleger [14].

Competing patterns extend the power of finite state transducer somewhat like adding the "not" operator to regular expressions.

Methods for the induction of covering patterns from W are needed.

Attempts to catch the regularities in empirical data (W in our case) can be traced back to the 1960s, when Chytil and Hájek started to generate unary hypotheses on finite models using the GUHA method [15].

Definition 9 (matrix representation of the data). *Let us have $m \times n$ matrix $W = w_{ij}$ of data that describe m objects with n binary attributes $P_1, P_2, \ldots, P_n$ (unary predicates). Either P_j or $\neg P_j$ holds.* Elementary conjunction *is a conjuction of literals P_j, $1 \leq j \leq n$, where every predicate appears once at most. Similarly,* Elementary disjunction *is a disjunction of literals P_j with the same condition. We say that the object i fulfills elementary conjunction Φ if the formula exactly describes the attributes in line i of W. We say that Φ holds for W if Φ holds for all objects (lines in W). We say that formula Φ is* p-truth *if Φ holds for at least $100p\%$ of objects, $p \in \mathbb{R}, 0 < p \leq 1$.*

We immediately see that we can represent our hyphenation problem by a matrix W: the attribute in column j, P_j tells whether a word can be divided (true or 1) or not (false or 0).

GUHA method searches for such elementary conjunctions A (*antecedents*) and elementary disjunctions S (*succedents*) with no common predicates, such that implication $A \rightarrow S$ is p-truth; it searches for hypotheses with highest p to detect dependencies in data. Observational language in this case is *propositional logic*. There are many general approaches using first-order predicate calculus or even higher formalisms [16], but these are not necessary for our task.

Definition 10 (p-truth pattern α). *Let us have m hyphenated words represented in matrix W as in Definition 9 on the facing page. We say that pattern α is p-*truth pattern *if it covers at least* $100p\%$ *of applicable word segmentation points.*

The greedy approach for pattern search consists in collecting p-truth patterns with the highest p of the shortest length. Short patterns give a high generalization and good minimalization of space for pattern storage. But during its generation some heuristics have to be used, as maximal coverage of covering patterns does not imply good performace in the succeeding phases of pattern generation (of inhibiting patterns). Further discussion on pattern preparation could be found in [13,17,3].

4 Methods of Competing Patterns Generation

The idea of competing patterns is taken from the method developed by Liang [13] for his English hyphenation algorithm. It has been shown by extensive studies [18,19,20] that the method scales well and that parameters of the pattern generator – PATGEN program [21] – could be fine-tuned so that virtually all hyphenation points are covered, leading to about 99.9% efficiency.

The methodology consists of several parts:

stratification – for repetitive pattern generation, it is practical to have a stratified word list with 'information bearing' samples only;

bootstrapping – input data (word list with marked hyphenation points) preparation;

goal-driven threshold setting heuristics – the quality of generated patterns depends on many parameters that have to be set in advance;

data filtering by threshold setting heuristics – we can filter out 'dangerous' data – data that are hard to learn for manual inspection.

4.1 Stratification

Word lists from which patterns are generated may be rather big. A full list of Czech word forms has about 6,000,000 entries when generated by the Czech morphological analyzers ajka or majka. It may be even more than that for other tasks with huge input data collections such as POS tagging, or Thai text segmentation [22]. Context necessary for ambiguity resolution is often repeated several times – a word list may be stratified. Stratification means that from 'equivalent' words only one or small number of representatives are chosen for the pattern generation process.

With the stratification procedure described in [19] we have downsampled 3,300,000 Czech word forms to a word list of 372,562 word forms (samples) for PATGEN input. The same approach was used also for Slovak.

Stratified sampling is less important when we insist on lossless compression, or when we have enough computing power for pattern generation.

4.2 Bootstrapping

The preparation of data for machine learning is often a time-consuming task and for extremely large data sets, a technique called *bootstrapping* is used. It was used for tagging the ORCHID corpus [22] and for tagging word divisions it is also usefull. The idea is to tag only small initial data set (word list), and generate patterns from this input. Then, these bootstrap patterns are used for the automatic tagging of a bigger input list, and checked before the next pattern generation phase.

Bootstrapping may bring errors especially with overlapping prefixes (ne-, nej-, po-, pod-). It is worth the trouble marking these points separately, e.g., with the help of a morphological analyzer.

4.3 Pattern Generation

Pattern generation processes are driven by several threshold parameters whose settings are essential for the quality and properties (precision and recall) of generated patterns. Our experience shows that parameter setting not only depends on the requested pattern behaviour but to a certain extent on the problem at hand. Parameter setting has to be tuned for every pattern generation project.

PATGEN runs at various levels. At every level, a new set of patterns is generated. It starts with short patterns (counting frequencies of substrings of a given length), generating longer ones in the next level as 'exceptions', and making 'exceptions of exceptions' in the next level, etc. With this model, we can learn exact dependencies between contexts of hyphenation points in words that are used in a much wider context than can standard (bi|tri)gram or other statistical methods taken into consideration – there are examples when the segmentation decision depends on the word segment that is six characters away.

There is no known algorithm that helps with setting of the parameters of the learning process. Liang's original patterns (`hyphen.tex`) that are in every TEX distribution as a default patterns for (American) English are very inefficient and have very low recall. They cover only 89.3% [13, page 37] – of very small word list (Webster's Pocket Dictionary) of 49,858 words. The threshold used in pattern generation were not tuned at all, and better choices can lead to smaller pattern size and higher (actually complete) coverage of hyphenation points in an input word list.

4.4 Pattern Parameter Setting Optimization

Our extensive experience shows that parameter setting is highly language dependent – it differs when generating patterns for Thai segmentation [22] for Czech and Slovak hyphenations [19] Scannel [23] reports that using this methodology he generated a new pattern for Irish that does not produce any hyphen points which are not in the database and miss just 10 out of 314,639

hyphen points. This is consistent with our findings that the methodology is usable as very effective lossless compression algorithm, and there is the power of competing patterns to cover *all* exceptions from data.

We may experiment with parameter setting so that generated patterns are *nearly* lossless. Words that were not covered in this phase are in some way different than the rest. This difference may well be right, but usually show an input data tagging error. We suggest manually checking this small set of words esspecially when developing and marking new word lists from scratch.

4.5 Layering of Disambiguation Patterns

There can be a different version of the input data (different variants of segmentation, tagging), with different patterns. As competing patterns are decomposable into layers, we can "plug-in" patterns developed by experts on the problem and merge or compare them with those generated. We can let the patterns "compete" – or adjust them so that, for example, expert knowledge takes preference over generated patterns, or we can take the expert patterns as initial set of patterns and generate the patterns to cover the rest of the input data. It has been shown [19] that hyphenation patterns were often done by hand, or by a combination of hand crafted and generated patterns. Having several layers of expert patterns, we can easily set up their priorities by changing the classification numbers in the patterns. This priority handling is necessary in most information fusion tasks.

5 Summary and Conclusions

In this paper, we have formally proved the hardness of creation of space-optimal competing patterns for segmentation tasks. Even though the theoretical result seems negative, in practical applications like hyphenation, we are still able to find space efficient patterns that are able to solve segmentation task in constant time, e.g. irrespectively of the number of segmented inputs.

Techniques like bootstrapping, stratification and parameter generation setting heuristics allows for efficient working with language data to be segmented – a work for language experts. Fixing errors in data then allows for better and smaller pattern sets that are close to the theoretical optimum.

This all opens new horizons on usage of competing patterns as a new compact data structure in many NLP tasks.

Acknowledgements This work has been partially supported by the European Union through its Competitiveness and Innovation Programme (Information and Communications Technologies Policy Support Programme, "Open access to scientific information", Grant Agreement No. 250503).

References

1. Hofstadter, D.R.: Gödel, Escher, Bach: An Eternal Golden Braid. Basic Books (1979)
2. Goldsmith, J.: Unsupervised Learning of the Morphology of a Natural Language. Computational Linguistics **27**(2) (2001) 153–198
3. Sojka, P.: Competing Patterns in Language Engineering and Computer Typesetting. PhD thesis, Masaryk University, Brno (2005)
4. Manning, C.: Part-of-Speech Tagging from 97% to 100%: Is It Time for Some Linguistics? In Gelbukh, A., ed.: Computational Linguistics and Intelligent Text Processing, 12th International Conference CICLing 2011, Part 1, LNCS 6608, Springer (2011) 171–189
5. Hofstadter, D.R.: Artificial intelligence: Subcognition as computation. (1983)
6. Brill, E., Florian, R., Henderson, J.C., Mangu, L.: Beyond N-Gram: Can Linguistic Sophistication Improve Language Modeling? In: Proceedings of the ACL '98. (1998)
7. Jiang, T., Salomaa, A., Salomaa, K., Yu, S.: Decision problems for patterns. Journal of Computer and Systems Sciences **50**(1) (1995) 53–63
8. Gross, M.: The Construction of Local Grammars. [24] 329–354
9. Hobbs, J.R., Appelt, D., Bear, J., Israel, D., Kameyama, M., Stickel, M., Tyson, M.: FASTUS: A Cascaded Finite-State Transducer for Extracting Information from Natural-Language Text. [24] 383–406
10. Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing. MIT Press (1999)
11. Ausiello, G., Gambosi, G., Crescenzi, P., Kann, V.: Complexity and Approximation. Springer-Verlag (1999)
12. Chvátal, V.: A Greedy Heuristic for the Set Covering Problem. Mathematics of Operations Research **4** (1979) 233–235
13. Liang, F.M.: Word Hy-phen-a-tion by Com-put-er. PhD thesis, Department of Computer Science, Stanford University (1983)
14. Pfleger, C.P.: State Reduction in Incompletely Specified Finite-State Machines. IEEE Trans. Computers C **22**(4) (1973) 1099–1102
15. Hájek, P., Havránek, T.: Mechanising hypothesis formation – Mathematical foundations for a general theory. Springer-Verlag (1978)
16. Lloyd, J.W.: Learning Comprehensible Theories from Structured Data. In Mendelson, S., Smola, A., eds.: Advanced Lectures on Machine Learning, LNAI 2600. (2003) 203–225
17. Sojka, P.: Competing Patterns for Language Engineering. In Sojka, P., Kopeček, I., Pala, K., eds.: Proceedings of the Third International Workshop on Text, Speech and Dialogue—TSD 2000. Lecture Notes in Artificial Intelligence LNCS/LNAI 1902, Brno, Czech Republic, Springer-Verlag (2000) 157–162
18. Sojka, P., Ševeček, P.: Hyphenation in TEX – Quo Vadis? TUGboat **16**(3) (1995) 280–289
19. Sojka, P.: Notes on Compound Word Hyphenation in TEX. TUGboat **16**(3) (1995) 290–297
20. Sojka, P.: Hyphenation on Demand. TUGboat **20**(3) (1999) 241–247
21. Liang, F.M., Breitenlohner, P.: `PAT`tern `GEN`eration program for the TEX82 hyphenator. Electronic documentation of `PATGEN` program version 2.3 from web2c distribution on CTAN (1999)
22. Sojka, P., Antoš, D.: Context Sensitive Pattern Based Segmentation: A Thai Challenge. In Hall, P., Rao, D.D., eds.: Proceedings of EACL 2003 Workshop on Computational Linguistics for South Asian Languages – Expanding Synergies with Europe, Budapest (2003) 65–72

23. Scannell, K.P.: Hyphenation patterns for minority languages. TUGboat **24**(2) (2003) 236–239
24. Roche, E., Schabes, Y.: Finite-State Language Processing. MIT Press (1997)

Author Index

RASLAN 2012

Sixth Workshop on Recent Advances in Slavonic Natural Language Processing

Editors: Aleš Horák, Pavel Rychlý
Typesetting: Adam Rambousek
Cover design: Petr Sojka

Printed and published by Tribun EU s. r. o.
Cejl 32, 602 00 Brno, Czech Republic

First edition at Tribun EU
Brno 2012

ISBN 978-80-263-0313-8

www.librix.eu