RASLAN 2008
Recent Advances in Slavonic Natural Language Processing

Masaryk University
http://nlp.fi.muni.cz/raslan/2008/

P. Sojka, A. Horák (Eds.)

RASLAN 2008

Recent Advances in Slavonic Natural Language Processing

Second Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2008
Karlova Studánka, Czech Republic, December 5–7, 2008
Proceedings

Masaryk University
Brno 2008

Proceedings Editors

Petr Sojka
Faculty of Informatics, Masaryk University
Department of Computer Graphics and Design
Botanická 68a
CZ-602 00 Brno, Czech Republic
Email: sojka@fi.muni.cz

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

Katalogizace v knize – Národní knihovna ČR

RASLAN 2008 (2. : Karlova Studánka, Česko)

RASLAN 2008 : Recent Advances in Slavonic Natural Language Processing : second workshop on ..., Karlova Studánka, Czech Republic, December 5-7, 2008 : proceeedings / P. Sojka, A. Horák (eds.). - 1st ed. - Brno : Masaryk University, 2008. - viii, 102 s.

ISBN 978-80-210-4741-9

81'322 * 004.82/.83:81'322.2
- computational linguistics
- natural language processing
- proceedings of conferences
- počítačová lingvistika
- zpracování přirozeného jazyka
- sborníky konferencí

410 - Linguistics [11]
006.3 - Artificial intelligence [23]

81 - Lingvistika. Jazyky [11]
004.8 - Umělá inteligence [23]

Printed in Czech Republic

ISBN 978-80-210-4741-9

Preface

This volume contains the Proceedings of the Second Workshop on Recent Advances in Slavonic Natural Language Processing (RASLAN 2008), organized by the the Center of Natural Language Processing at the Faculty of Informatics, Masaryk University and held on December 5th–7th 2008 in Karlova Studánka, Kurzovní chata, Jeseníky, Czech Republic.

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

Topics of the Workshop include (but are not limited to):

* text corpora and tagging
* syntactic parsing
* sense disambiguation
* machine translation, computer lexicography
* semantic networks and ontologies
* semantic web
* knowledge representation
* applied systems and software for NLP

RASLAN 2008 offers a rich program of presentations, short talks, technical papers and mainly discussions. A total of 16 papers and abstracts were accepted, contributed altogether by 23 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 Pavel Rychlý, Aleš Horák and Dana Hlaváčková. The TEXpertise of 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 Masaryk University as publisher of these proceedings, and of tribun.eu as printer is gratefully acknowledged.

Brno, November 2008 Karel Pala

Table of Contents

Towards Czech Morphological Guesser

Pavel Šmerk

Faculty of Informatics, Masaryk University
Botanická 68a, CZ-602 00 Brno, Czech Republic
smerk@mail.muni.cz

Abstract. This paper presents a morphological guesser for Czech based on data from Czech morphological analyzer `ajka` [1]. The idea behind the presented concept lies in a presumption that the new (and therefore unknown to the analyzer) words in a language behave quite regularly and that a description of this regular behaviour can be extracted from the existing data of the morphological analyzer. The paper describes both the construction of guesser data and the architecture of the guesser itself.

1 Introduction

An obvious disadvantage of traditional morphological analyzers is the finiteness of their dictionaries. There is no way to catch *all* words of the particular language in a dictionary of an analyzer, because new and new words continue to appear in the language. Thus, almost allways there will be some words on which the analyzer will not be able to return any information.

If we want to process even these words unrecognized by the analyzer, we have two possibilities. Either to guess possible lemma (or lemmata) and appropriate tags from the context of the word, or to guess it from the form of the word, i. e. from its resemblance to some word known to the analyzer.

In this paper we describe the second of these two possible approaches. First of all, a general idea will be introduced. In the next section, we describe the algorithm for construction of the data as well as the architecture of the guesser in section 4. At the end we make some remarks on possible future development of a guesser.

2 General Idea

The Czech language has many inflectional and derivational paradigms (especially if we count every “exception" as a separate paradigm, as the morphological analyzer `ajka` does), but only a smaller part of them is synchronically productive in the sense that there are appearing (or at least may appear) new words in the language which belong to that productive paradigms.

For instance, word *husa* (“goose") has an own, separate paradigm, because of a doublet in genitive plural, *hus* and *husí*. There is no other word with the same inflectional paradigm and no chance that any such word could appear in

Petr Sojka, Aleš Horák (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2008, pp. 1–4, 2008.

Czech. All new feminines which end with vocal *-a* will belong to the paradigm *žena* (“woman").

Of course, it is similar for the derivational relations. For instance, some older adjectives with a meaning “made of material/substance" are created with suffix *-en(ý)*, e. g. *dřevěný* (“wooden”). But nowadays this suffix is not productive and this meaning is expressed by productive suffix *-ov(ý)*.

Obviously, the non-productive paradigms are quite useless for the guessing the possible lemma and tags, or even more than that: these paradigms might serve as an undesirable bias if we would not ignore them. Unfortunately, we do not have the information, which paradigms are productive and which are not.

As was said above, we may assume that all Czech words unrecognized by the analyzer are regular, i. e. belong to some productive paradigm. Or, in other words, that the words with irregular behaviour are either already known to the analyzer or they are quite infrequent in real texts and thus irrelevant for our task. Moreover, we may assume that absolute majority of the actually used lexicon is covered by the dictionary of the analyzer, which means that there is enough examples of all productive behaviour in that dictionary. Then we may conclude that “productive" significantly correlates with “frequent in existing data" and our goal will be to sort out the frequent behaviour from the analyzer’s dictionary.

3 Construction of the Guesser Data

In this section, the algorithm of the guesser data construction will be described in detail.

- First of all, we count numbers of lemmata for each paradigm from the dictionary of Czech morphological analyzer `ajka`,
- then we pick out all lemmata (from that dictionary) which belong to any paradigm which has at least 20 lemmata (this will reduce the number of lemmata from 389,831 to 382,994, i. e. by 1.75 %, but the number of paradigms from 1,830 to 387, i. e. by 78.85 %),
- we let the morphology analyzer generate all word forms for each lemma picked out in the previous step, if the lemma belongs to some of the open POS categories (i. e. nouns, adjectives, verbs or adverbs — other POS categories are closed, so we need not expect any new words which would belong to them),
- moreover we discard all colloquial word forms, all negations and superlatives and all word forms of length 6 letters or less,
- each word form we turn into a triplet word form, lemma and tag in the following form: `akzeřuo:ka:ek:k1gMnSc2,k1gMnSc4` where the `akzeřuo` is the reversed (letter by letter) word form *ouřezka*, the next two strings `ka` and `ek` specifies a lemma (one has to remove the first string from the end of the word form and append the second string, i. e. lemma in the example is *ouřezek*) and the last is a list of possible morphological tags,

– we sort the triplets lexicographically (the actual sort order is not important),
– then we go through the list of triplets and collect information on possible word forms' ends of various length and corresponding lemmata and tags:
  - we use two arrays, both of length 10. The elements of the first array represent letters from the end of the processed word form so that each left "subarray" represents the end of the word form of the given length. The elements of the second array are hashes, in which the keys are triplets without the word form (e. g. `ka:ek:k1gMnSc2,k1gMnSc4`) and values indicates for how many word forms (from the list of triplets) with the given end the analyzer returns this lemma and tag(s),
  - it would be difficult to describe the following steps in general, that is why we illustrate it on example,
  - let us suppose that after the processing of the triplet `akzeřuo:ka:ek:k1gMnSc2,k1gMnSc4`, the content of the first array is

1	2	3	4	5	6	7	8	9	10
a	k	z	e	-	-	-	-	-	-

    (which means that the first hash from the second array stores possible interpretations [lemma and tag(s)] of words which end with *-a*, the second hash is information on words which end with *-ka* etc.),
  - and that the next two triplets are `akzeřýv:ka:ek:k1gMnSc2,k1gMnSc4` and `akzjapš:::k1gFnSc1`,
  - we assume that at least the first (or the last, in reversed string) three letters represent the root of the word form, that is, none of these letters are part of any derivational suffix of flectional ending and thus we can (and should) ignore them. In addition, we assume, that eventual combinations of suffixes and ending longer than 10 letters are not interesting for the guesser,
  - it follows that in our example we ignore *ouř*, *výř* and *špa* and take only the ends *ezka* (*akze*) and *jzka* (*akzj*),
  - for each of such ends we compare its letters with the letters in the first array. We do it from the highest index (10) to the lowest (1). There are three possibilities[1]:
    * both positions are empty ("-" signs emptiness). We do nothing,
    * both positions are nonempty, but equal. We take the rest of the triplet (e. g. `ka:ek:k1gMnSc2,k1gMnSc4`) as a key in the corresponding hash in the second array and increase the associated value,
    * positions are different. We empty the corresponding hash and replace the content of the array element with the letter from the end of the word. Before emptying the hash, we store the information it contains, but only if total sum of values exceeds 20 and at least one value exceeds 10. In such case we generate a string by joining the following data with some separator: given end of the word form, total sum of values and for each key its value and the key itself.

[1] In fact, things are even a little bit more complicated.

Moreover, all numbers, which we store in the generated string, has to be subtracted from the corresponding values in all hashes with lower index than the current hash has,

- back to our example: the word end *akze* (word form *výřezka*) is the same as the letters in the first array, so we only update the first four hashes by adding 1 to the value associated with key `ka:ek:k1gMnSc2,k1gMnSc4`. But the next word end *akzj* (word form *špajzka*) differs at the fourth position. Therefore we empty the hash, generate something like `akze 50 30:ka:ek:k1gMnSc2,k1gMnSc4 20:::k1gFnSc1` (just an example, not the real data) and put *j* into the fourth element of the first array,

– we sort the generated strings lexicographically, do some optimizations (merge some very similar data etc.), strip the numbers, reverse the sort order and the data for the guesser are done.

4 Guesser Architecture

The guesser itself is rather simple. During the start, it reads the data and creates a regular expression from word ends and a hash, in which the keys are the word ends and values are the information about possible lemmata and tags. Words, which are not recognized by the morphological analyzer, are reversed and their last three letters are cut off. The rest is matched against the regular expression, the eventual match (which is always the longest possible one, because of reversed sort order of the data) is found in the hash and the corresponding value is sent to the output.

5 Future Development

The main problem of this approach is that it works well only for word forms which have some derivational suffixes or inflectional endings, i. e. which are integrated into the Czech language system. It will not success in guessing indeclinable words such as expressions from other languages, foreign proper names etc. It seems that proper handling of these words will require some contextual information.

Acknowledgements. This work has been partly supported by the Academy of Sciences of Czech Republic under the project 1ET200610406, by the Ministry of Education of CR within the Center of Computational Linguistics LC536, by the Czech Science Foundation under the project GA407/07/0679, and in the National Research Programme II project 2C06009.

References

1. Radek Sedláček and Pavel Smrž. 2001. *A New Czech Morphological Analyser* `ajka`. In Proceedings of the 4th International Conference TSD 2001. LNCS 2166, Springer-Verlag, pp. 100–107.

Semi-automatic Approach to Building Dictionary between Slavonic Languages
Abstract

Marek Grác

Faculty of Informatics, Masaryk University
Botanická 68a, 60200 Brno, Czech Republic
xgrac@fi.muni.cz

Abstract. Machine translation between Slavonic languages is still in its early stages. Existence of bilingual dictionaries have big impact on quality of translation. Unfortunately creating such language resources is quite expensive. For small languages like Czech, Slovak or Slovenian is almost sure that large-enough dictionary will not be commercially successful. Slavonic languages tends to range between close and very close languages so it is possible to infer some translation pairs. Our presentation focus on describing semi-automatic approach using 'cheap' resources for Czech-Slovak and Serbian-Slovenian dictionary. These resources are stacked so in earlier phases we will receive results of higher precision. Our results show that this approach improves effectivity of building dictionaries for close languages.

Petr Sojka, Aleš Horák (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2008, pp. 5–5, 2008.

A Lexicographer-Friendly Association Score

Pavel Rychlý

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

Abstract. Finding collocation candidates is one of the most important and widely used feature of corpus linguistics tools. There are many statistical association measures used to identify good collocations. Most of these measures define a formula of a association score which indicates amount of statistical association between two words. The score is computed for all possible word pairs and the word pairs with the highest score are presented as collocation candidates. The same scores are used in many other algorithms in corpus linguistics.
The score values are usually meaningless and corpus specific, they cannot be used to compare words (or word pairs) of different corpora. But end-users want an interpretation of such scores and want a score's stability. This paper present a modification of a well known association score which has a reasonable interpretation and other good features.

1 Introduction

Finding collocation candidates is one of the most important and widely used feature of corpus linguistics tools [1]. There are many statistical association measures used to identify good collocations. Most of these measures define a formula of a association score which indicates amount of statistical association between two words. The score is computed for all possible word pairs and the word pairs with the highest score are presented as collocation candidates. The same scores are used in many other algorithms in corpus linguistics, for example to compute collocations in grammatical relations and an importance of grammatical relations in the Sketch Engine [2].

There are two general problems of most association scores:

1. A score is fine-tuned to one particular corpus size and/or key word frequency. If we use a score for a corpus with very different number of tokens the resulting list is not satisfying enough or is completely wrong.
2. The score values are usually meaningless and corpus specific, they cannot be used to compare words (or word pairs) of different corpora. But end-users want an interpretation of such scores and want a score's stability. They want to compare collocation scores of different words and on different corpora or subcorpora.

Petr Sojka, Aleš Horák (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2008, pp. 6–9, 2008.

The article is organized as follows. The following section describe notation and the most widely used association scores. The Section 3 illustrates these two problems on real examples. The next section defines a new score *logDice*, which is a modification of the well known association score *Dice* [3]. The *logDice* score has a reasonable interpretation, scales well on a different corpus size, is stable on subcorpora, and the values are in reasonable range.

2 Association Scores for Collocations

Almost all association score formulas use frequency characteristics from a contingency table, which records the relationship between two words (W_1, W_2). Table 1 shows an example of a contingency table. The numbers in the right-hand column and the bottom row are called marginal frequencies and the number in the bottom right-hand corner is the size of the corpus.

In the rest of this paper we will uses the following symbols (the meaning is also summarized in Table 1):

- f_x = number of occurrences of word X
- f_y = number of occurrences of word Y
- f_{xy} = number of co-occurrences of words X and Y
- $R_x = \frac{f_{xy}}{f_x}$ = relative frequency of word X
- $R_y = \frac{f_{xy}}{f_y}$ = relative frequency of word Y

Table 1. Notation of frequencies of words X and Y

	$W_1 = X$	$W_1 \neq X$	
$W_2 = Y$	f_{xy}	$f_y - f_{xy}$	f_y
$W_2 \neq Y$	$f_x - f_{xy}$	$N - f_{xy}$	$N - f_y$
	f_x	$N - f_x$	N

3 Widely Used Association Scores

This section summarize formulas of some association scores and gives its main characteristics. More scores, motivations, discussion of their mathematical background and full references could be find in [4].

T-score: $\frac{f_{xy} - \frac{f_y f_B}{N}}{\sqrt{f_{xy}}}$

MI-score: $\log_2 \frac{f_{xy} N}{f_x f_y}$

MI^3-score: $\log_2 \frac{f_{xy}^3 N}{f_x f_y}$

Minimum Sensitivity: $\min R_x, R_y$

Dice coefficient: $D = \frac{2f_{xy}}{f_x + f_y}$

MI log Freq: *MI-score* $\times \log f_{xy}$, used as salience in the first version of Word Sketches [2].

Table 2 lists the collocation candidates on lemmas to the verb *break* in the window from 5 tokens to the left to 5 tokens to the right. They were computed on the British National Corpus by the Manatee system [5].

Table 2. Collocation lists for different association scores

	F_{xy}	**T-score**		F_{xy}	**MI-score**		F_{xy}	**MI^3-score**
the	11781	99.223	spell-wall	5	11.698	the	11781	30.591
.	8545	83.897	deadlock	84	10.559	down	2472	29.882
,	8020	80.169	hoodoo	3	10.430	.	8545	29.558
be	6122	69.439	scapulum	3	10.324	,	8020	29.193
and	5183	65.918	Yasa	7	10.266	be	6122	28.311
to	5131	65.798	intervenien	4	10.224	to	5131	28.268
a	3404	52.214	preparedness	21	10.183	and	5183	28.246
of	3382	49.851	stranglehold	18	10.177	into	1856	27.854
down	2472	49.412	logjam	3	10.131	up	1584	26.967
have	2813	48.891	irretrievably	12	10.043	a	3404	26.717
in	2807	47.157	Andernesse	3	10.043	have	2813	26.593
it	2215	43.314	irreparably	4	10.022	of	3382	26.255
into	1856	42.469	Thief	37	9.994	in	2807	26.095
he	1811	39.434	THIEf	4	9.902	it	2215	25.876
up	1584	39.038	non-work	3	9.809	out	1141	25.821

	F_{xy}	**Min. Sens.**		F_{xy}	**MI log Freq**		F_{xy}	**Dice**
down	2472	0.027	down	2472	57.340	down	2472	0.0449
silence	327	0.018	silence	327	48.589	silence	327	0.0267
leg	304	0.016	deadlock	84	46.909	into	1856	0.0210
law	437	0.014	barrier	207	46.389	leg	304	0.0203
heart	259	0.014	into	1856	46.197	off	869	0.0201
rule	292	0.013	off	869	42.411	barrier	207	0.0191
off	869	0.013	up	1584	42.060	law	437	0.0174
news	236	0.013	leg	304	41.980	up	1584	0.0158
into	1856	0.012	neck	180	39.336	heart	259	0.0155
barrier	207	0.011	law	437	38.805	neck	180	0.0148
away from	202	0.011	out	1141	38.783	news	236	0.0144
war	294	0.010	bone	151	38.263	rule	292	0.0142
ground	182	0.010	heart	259	37.327	out	1141	0.0135
record	287	0.010	Thief	37	36.353	away from	202	0.0135
neck	180	0.010	news	236	36.296	bone	151	0.0130

4 logDice

As one can see from the previous section, *Dice* score gives very good results of collocation candidates. The only problem is that the values of the *Dice* score are usually very small numbers. We have defined *logDice* to fix this problem.

$$logDice = 14 + \log_2 D = 14 + \log_2 \frac{2f_{xy}}{f_x + f_y}$$

Values of the *logDice* have the following features:

- Theoretical maximum is 14, in case when all occurrences of X co-occur with Y and all occurrences of Y co-occur with X. Usually the value is less then 10.
- Value 0 means there is less than 1 co-occurrence of XY per 16,000 X or 16,000 Y. We can say that negative values means there is no statistical significance of XY collocation.
- Comparing two scores, plus 1 point means twice as often collocation, plus 7 points means roughly 100 times frequent collocation.
- The score does not depend on the total size of a corpus. The score combine relative frequencies of XY in relation to X and Y.

All these characteristics are useful orientation points for any field linguist working with collocation candidate lists.

5 Conclusion

In this paper, we have presented the new association score *logDice*. The *logDice* score has a reasonable interpretation, scales well on a different corpus size, is stable on subcorpora, and the values are in reasonable range.

Acknowledgments. This work has been partly supported by the Academy of Sciences of Czech Republic under the projects 1ET200610406, 1ET100300419 and by the Ministry of Education of CR within the Centre of basic research LC536 and National Research Programme 2C06009.

References

1. Smadja, F.: Retrieving Collocations from Text: Xtract. Computational Linguistics **19**(1) (1994) 143–177.
2. Kilgarriff, A., Rychlý, P., Smrž, P., Tugwell, D.: The Sketch Engine. Proceedings of Euralex (2004) 105–116.
3. Dice, L.: Measures of the Amount of Ecologic Association Between Species. Ecology **26**(3) (1945) 297–302.
4. Evert, S.: The Statistics of Word Cooccurrences: Word Pairs and Collocations. Unpublished Ph.D. dissertation, University of Stuttgart (2004).
5. Rychlý, P.: Manatee/Bonito – A Modular Corpus Manager. In: P. Sojka, A. Horák (Eds.): RASLAN 2007 Proceedings (2007), pp. 65–70.

Corpus Architect
Abstract

Jan Pomikálek

Faculty of Informatics, Masaryk University, Brno, Czech Republic
xpomikal@fi.muni.cz

Abstract. In the Corpus Architect project we are developing a system for creating text corpora, which will be easy to use even for non-technical users. The system creates corpora from two data sources – users' own text documents and web pages. Users can upload their texts in various formats or ask the system for adding domain specific web texts into their corpora in an automated way. In the latter case the domain of the web texts is defined in terms of key words.

Once all the desired texts are collected for the corpus, with a single click users can have the texts part-of-speech tagged, lemmatized and loaded into the Sketch Engine corpus manager. With the corpus manager, users can instantly make use of fast searching in the corpus. They also immediately get access to important corpus derived statistical information, such as word sketches and statistical thesaurus.

The interface of the Corpus Architect is designed with an emphasis on simplicity and easiness of use. The user is asked for as little input as possible. No decisions are requested for options if the value can be detected in an automated way or a reasonable default can be used.

A live demo of the system will be presented. However, as long as this is a work in progress and many features are still unimplemented, only the basic functionality will be shown.

Petr Sojka, Aleš Horák (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2008, pp. 10–10, 2008.

The Saara Framework
Work in Progress

Vašek Němčík

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

Abstract. The determination of reference and referential links in discourse is one of the important challenges in natural language understanding. The first commonly adopted step towards this objective is to determine coreference classes over the set of referring expressions. We present a modular framework for automatic anaphora resolution which makes it possible to specify various anaphora resolution algorithms and to use them to build AR systems, in principle, for any natural language. The functionality of the system is shown on selected salience-based algorithms customized for Czech.

1 Introduction

In this work, we present Saara (System for Automatic Anaphora Resolution and Analysis), a framework for anaphora resolution (AR) which is modular in many ways. Modularity in the context of AR has many obvious advantages. It allows easy experimentation on various algorithms, their evaluation using various metrics, and easy use of already implemented algorithms with various languages and data formats. In our framework, this was achieved mainly by defining multiple abstraction levels and separate modules for individual phases of processing. The architecture is in accord with the principles formulated by Byron and Tetreault [1] for their own system.

Space and time constraints do not permit an investigation across the whole spectrum of AR algorithm types. We chose to focus on traditional algorithms based on the concept of salience. Salience-based algorithms serve as a sensible initial probe for a language and at the same time provide a good basis for further implementations exploiting more complex resources. We evaluate the performance of these approaches for Czech.

At the moment, we decided to disregard grammatical coreference and rather concentrate on textual anaphora. Whereas grammatical coreference phenomena are typically subject to language-specific constraints applicable to particular syntactic constructions, textual reference phenomena tend to follow similar principles across languages, and therefore it makes sense to experiment with cross-linguistic reuse of algorithms.

Petr Sojka, Aleš Horák (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2008, pp. 11–16, 2008.

In the next section, we briefly review other modular AR systems and AR systems for Czech proposed in existing literature. Section 3 describes the architecture of our framework, sketches the algorithms re-implemented and provides evaluation figures. Finally, the last section suggests directions for future work.

2 Related Work

At present, mechanisms for performing anaphora resolution are becoming integral parts of modern NLP systems. Also for Czech, several AR algorithms have been formulated (e.g. [2,3,4]), however, the efforts to implement them have been substantially discouraged by the lack of Czech data suitable for evaluating AR.

The situation has changed only recently through the emergence of the Prague Dependency TreeBank [5,6], which contains annotation of pronominal coreference on its tectogrammatical level of representation. The Prague Dependency TreeBank (PDT), in its version 2.0, contains tree-representations of about 50,000 sentences with approximately 45,000 manually annotated coreference links (over 23,000 grammatical ones, and over 22,000 textual ones).

The only other relevant AR system for Czech known to me at this moment, was presented in the master thesis of Linh [7]. Her system, called AČA, contains a rule-based algorithm for resolving pronominal anaphors and defines machine-learning algorithms for all types of anaphors annotated in PDT 2.0. In our opinion, the only rather minor flaw that can be pointed out is the lack of detail concerning the presented evaluation figures.

The figures given in [7] are rather difficult to compare with the figures for our system presented below. Firstly, in AČA (unlike in Saara), the annotation data is used to detect the anaphors. Secondly, it treats nodes in certain artificially generated constructions as textual anaphora, whereas in our system, we exclude them either as nodes of technical character, or beyond the range of the AR task at the analysis depth we aim at.

To avoid problems of this sort, we took inspiration from earlier modular AR systems and their advantages. Here we mention at least the two which we consider most notable.

The first one was developed at the University of Rochester by Byron and Tetreault [1]. The authors emphasize the advantages of modularity and encapsulation of the system modules into layers. For their system, they define three: the AR layer containing functions addressing AR itself, the translation layer for creating data structures, and the supervisor layer for controlling the previous layers.

Another modular system was produced by Cristea et al. [8] and defines layers from a different perspective. The representation of the discourse being processed is divided into the text layer, containing the representation of the individual referring expressions, the projection layer, consisting of feature structures with attribute values describing the individual referring expressions,

and finally the semantic layer with representations of the individual discourse entities.

We agree with the authors of the above-mentioned frameworks that modularity is an invaluable characteristic in the context of AR, and we have put emphasis on this fact when laying the foundations for our own framework, described in the following section.

3 Saara and the Algorithms Re-implemented

In this section, we briefly describe the main features of our modular AR framework, sketch the algorithms used, and provide evaluation figures for them.

The main aspects of modularity reside in encapsulation of the AR algorithms. The encapsulation is achieved by defining two processing levels: markable[1] level and sentence structure level. All AR algorithms are formulated on the markable level, and thus abstract away from the actual formalism and data format used. Markable features and relations among them are accessible only through an interface that "translates" the concept in question to the actual sentence representation.

In other words, for each new data format, it is necessary to define methods determining how to recognize referential expressions, anaphors and important relationships between them (e.g. morphological agreement, embededness, the syntactic role in the current clause). Then, in principle, any AR algorithm implemented can be used with this data.

As already mentioned, we investigated classic (mainly salience-based) AR algorithms dealing with textual pronominal anaphora and compared their performance on Czech – using the annotation in PDT 2.0 as golden standard. The following algorithms have been re-implemented:

Plain Recency As a baseline, we consider an algorithm based on plain recency, which links each anaphor to the closest antecedent candidate agreeing in morphology.

The Hobbs' Syntactic Search [9] is one of the earliest yet still popular AR approaches. Unlike all other approaches mentioned here, it does not build any consecutive discourse model. It is formulated procedurally, as a search for the antecedent by traversing the corresponding syntactic tree(s). The traversal is specified by a number of straightforward rules motivated by research in transformational grammar. In spite of the fact that the underlying ideas are quite simple, the algorithm accounts for numerous common instances and its performance on English is even today regarded as respectable.

The BFP Algorithm [10] employs the principles of centering theory, a more complex theory for modeling local coherence of discourse. One of its main claims is that each utterance has a single center of attention. Further it postulates certain rules and preferences on how centers can be realized, referred

[1] Markable is a collection of sentence representation tokens that correspond to phrases that are either themselves anaphors, or have the potential to be their antecedents.

Table 1. Performance of the system in traditional measures

	Recency	Haj87	HHS95	Hobbs	BFP	L&L
`Classic`						
Precision	34.21	33.91	33.98	26.76	53.36	43.12
Recall	33.70	33.41	33.48	26.30	39.90	42.18
F-measure	33.95	33.66	33.72	26.53	45.66	42.64
Success rate	36.79	36.47	36.55	28.71	43.56	46.05
`MUC-6`						
Precision	41.78	41.33	41.33	38.87	52.26	49.86
Recall	37.28	36.81	36.80	33.91	39.20	46.28

to etc. These rules account for numerous phenomena concerning anaphors, such as certain garden-path effects. The BFP algorithm considers all possible referential linking combinations between two neighbouring utterances and applies the claims of centering theory to rule out the implausible ones and subsequently to select the most preferred one among those left.

Activation models considering TFA[2] were originally formulated in the Praguian framework of Functional Generative Description. It stipulates that the hearer and speaker co-operate during communication to build a common structure, the so-called Stock of Shared Knowledge (SSK), which among other things, reflects the fact that some entities previously mentioned in the discourse are more activated, i.e. closer to the attention of the hearer, than others. Hajičová [2] presented a simple model of SSK and a set of rules for updating it based on the current utterance and its topic-focus articulation (TFA). These rules are applied iteratively to preserve the correctness of the model at each point of the discourse. An anaphor is linked to the most activated item of the SSK agreeing in morphology. Hajičová, Hoskovec, and Sgall [4] extended the previous model by defining a more fine-grained activation scale and certain referential constraints.

The method of combining salience factors is inspired by the RAP system presented by Lappin and Leass [11]. Its main idea is that the salience of a discourse object is influenced by a variety of factors. Each of them contributes to its salience in an uniform way and can be attributed to certain well-defined features of the corresponding referring expression and its context. For example, one factor "rewards" entities mentioned in the subject position. With each new sentence, the salience values of all previously considered entities are cut by half to account for salience fading through time. The antecedent of an anaphor is identified as the most salient object according to the adopted factors.

The evaluation figures for the above-mentioned AR algorithms within Saara are given in Table 1. The presented results offer an exceptional opportunity to compare the individual algorithms. It is always difficult to compare results given by different authors, or obtained through different systems, as it is

[2] TFA stands for Topic-focus articulation; similar ideas are also known as information structure, or functional sentence perspective.

usually not clear what exactly has been counted and taken into account, and whether the choice of the data set used for evaluation did play a role. In contrast, the figures provided here offer a basis for a straightforward comparison, as they were acquired using the same pre-processing facilities, the same linguistic assumptions and the same data.

4 Future Work

The main goal of our work is to arrive at a modular, state-of-the-art AR system for Czech that would be readily used as a module in bigger NLP applications. With regard to the figures presented in the previous section, it is obvious that the precision of the algorithms needs to be improved. We would like to pursue several means of achieving that.

Firstly, as error analysis revealed this to be an interesting point, we plan to investigate the referential properties of Czech anaphors across clauses of complex sentences. Next, we are interested in the role of TFA in anaphorical relations. In spite of the abundant theoretical studies, practical experiments haven't confirmed any relevant theoretical claims based on TFA values. Finally, we aim at improving the quality of the AR process by exploiting linguistic resources available for Czech, especially the Verbalex valency lexicon [12] and Czech WordNet [13].

Acknowledgments. This work has been partly supported by the Ministry of Education of CR within the Center of basic research LC536 and in the National Research Programme II project 2C06009.

References

1. Byron, D.K., Tetreault, J.R.: A flexible architecture for reference resolution. In: Proceedings of the Ninth Conference of the European Chapter of the Association for Computational Linguistics (EACL-'99). (1999).
2. Hajičová, E.: Focusing – a meeting point of linguistics and artificial intelligence. In: Jorrand, P., Sgurev, V. (Eds.): Artificial Intelligence Vol. II: Methodology, Systems, Applications. Elsevier Science Publishers, Amsterdam (1987), pp. 311–321.
3. Hajičová, E., Kuboň, P., Kuboň, V.: Hierarchy of salience and discourse analysis and production. In: Proceedings of Coling '90, Helsinki (1990).
4. Hajičová, E., Hoskovec, T., Sgall, P.: Discourse modelling based on hierarchy of salience. The Prague Bulletin of Mathematical Linguistics (64) (1995), pp. 5–24.
5. Hajič, J., et al.: The Prague Dependency Treebank 2.0. Developed at the Institute of Formal and Applied Linguistics, Charles University in Prague. (2005) `http://ufal.mff.cuni.cz/pdt2.0/`.
6. Kučová, L., Kolářová, V., Žabokrtský, Z., Pajas, P., Čulo, O.: Anotování koreference v pražském závislostním korpusu. Technical report, Charles University, Prague (2003).
7. Linh, N.G.: Návrh souboru pravidel pro analýzu anafor v českém jazyce. Master's thesis, Charles University, Faculty of Mathematics and Physics, Prague (2006).

8. Cristea, D., Postolache, O.D., Dima, G.E., Barbu, C.: AR-engine – a framework for unrestricted co-reference resolution. In: Proceedings of The Third International Conference on Language Resources and Evaluation, LREC 2002, Las Palmas, Spain (2002).
9. Hobbs, J.R.: Resolving pronoun references. In: Grosz, B.J., Spärck-Jones, K., Webber, B.L. (Eds.): Readings in Natural Language Processing. Morgan Kaufmann Publishers, Los Altos (1978), pp. 339–352.
10. Brennan, S.E., Friedman, M.W., rd, C.J.P.: A centering approach to pronouns. In: Proceedings of the 25th Annual Meeting of the ACL, Stanford (1987), pp. 155–162.
11. Lappin, S., Leass, H.J.: An algorithm for pronominal anaphora resolution. Computational Linguistics **20**(4) (1994), pp. 535–561.
12. Hlaváčková, D., Horák, A.: VerbaLex – New Comprehensive Lexicon of Verb Valencies for Czech. In: Computer Treatment of Slavic and East European Languages, Bratislava, Slovakia, Slovenský národný korpus (2006), pp. 107–115.
13. Pala, K., Smrž, P.: Building Czech Wordnet. Romanian Journal of Information Science and Technology **7**(2–3) (2004), pp. 79–88.

TIL and Logic Programming

Martina Číhalová, Nikola Ciprich, Marie Duží, Marek Menšík

VŠB-Technical University Ostrava
17. listopadu 15, 708 33 Ostrava, Czech Republic
m.tina.cihal@gmail.com, nikola.ciprich@linuxbox.cz,
marie.duzi@vsb.cz, mensikm@gmail.com

Abstract. The paper introduces a method of transition from TIL into Prolog system and vice versa, in order to utilize Prolog inference machine in the deductive system of TIL. We specify a subset of the set of TIL constructions the elements of which can be encoded in Prolog language, and introduce the method of translation from TIL into Prolog. Since Prolog is less expressive than TIL, we have to build up a TIL functional overlay that makes it possible to realize the reverse transition from Prolog into TIL in a near to equivalent way.

Key words: TIL, $\mathcal{T}IL$-`Script` language, inference machine, Prolog

1 Introduction

Transparent Intensional Logic (TIL) is a highly expressive logical system apt for the logical analysis of natural language.[1] In our project 'Logic and Artificial Intelligence for Multi-agent Systems' we deal with the problem of agents' communication with each other as well as with their environment. Human-computer communication should be smooth and near to isomorphic to natural-language communication. For this reason we voted for the language of TIL *constructions* as a specification and communication language of a multi-agent system. We develop the $\mathcal{T}IL$-`Script` language, a computational variant of TIL, which serves as a content language of agents' messaging.[2] $\mathcal{T}IL$-`Script` is a TIL dialect using only ASCII characters and slightly adjusted semantics. On the other hand, a great expressive power is inversely proportional to an easy implementation of a suitable automatic deductive system. Since TIL is a logic based on the infinite ramified hierarchy of types, it is impossible to create a complete logical calculus and make use of a standard automatic theorem prover. Though TIL deductive system has been theoretically specified, its implementation is still a work in progress. For this reason we decided to specify a subclass of TIL and to utilize a first-order inference machine such as Prolog. This is a feasible way due to TIL being a fully compositional system and from this point of view extensional calculus.

[1] See, for instance, [7] and [8]. [2] For details on the project, see [4]. $\mathcal{T}IL$-`Script` and agents' messaging is FIPA compliant. For FIPA standards see [6]. More on $\mathcal{T}IL$-`Script` , see [2] or [3].

Petr Sojka, Aleš Horák (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2008, pp. 17–30, 2008.

The goal of this paper is to specify a subset of TIL constructions which can be encoded in Prolog, together with the algorithm of their translation. The transition from TIL into Prolog and *vice versa* has to be specified in a near to equivalent way, i.e., without the loss of important information encoded by TIL constructions. Thus prior to the translation proper, TIL functional overlay of the Prolog machine has to be specified. In particular, we have to take into account that standard Prolog does not work with modal and temporal parameters, and with the types of denoted entities. Moreover, $\mathcal{TIL}$-`Script` is a functional language whereas Prolog a relational one, which is also a problem that has to be dealt with prior to translation proper. In the paper we propose a general strategy of adjusting TIL constructions into the form that can be processed by Prolog.

The paper is organized as follows. Section 2 briefly describes basic principles of programming in logic using Prolog. The main Section 3 is a description of the transition from TIL into Prolog. We first describe the translation of simple sentences into Prolog facts, rules and goals. Then we specify the method of translating wh-questions and yes-no questions. Finally, an algorithm of the specification of TIL subset of constructions transferable into Prolog is presented. Concluding Section 4 is an outline of future research.

2 A Brief Description of the Prolog System

Prolog can be defined as a couple (K, I), where K is the program base consisting of facts, rules and queries encoded in a first-order language, and I is an inference machine that makes it possible to derive consequences of K.[3] The inference machine is based on the general resolution algorithm for the first-order predicate logic (FOL). An input for the resolution algorithm is a set of formulas in the Skolem clausal form. Since FOL is only partly decidable,[4] the clauses of a Prolog language are limited to the decidable subset of FOL, namely the set of *Horn clauses*. To put this description on a more solid ground, we define:

Literal is an atomic formula or its negation. For instance, $p(x, y)$, $\neg q(f(x))$ are literals. *Clause* is a disjunstion of (positive and/or negative) literals:

$$C = l_1 \vee l_2 \vee \ldots \vee l_n \vee \neg m_1 \vee \ldots \vee \neg m_k\,.$$

Skolem clausal from of a formula is a conjunctive normal form without existential quantifiers

$$SCF = \forall x_1 \ldots \forall x_m (C_1 \wedge \ldots \wedge C_n),$$

where $C_1, \ldots, C_n$ are clauses.

[3] For details on Prolog see, e.g. [1]. [4] As a consequence of Gödel's incompleteness theorem, there is no proof calculus deciding FOL. Church proved that there are calculi that partly decide FOL; this means that if a formula F is logically valid then it is provable, whereas if F is not logically valid then it can be the case that there is no finite proof of F in the calculus.

Each FOL formula F can be transformed into *SCF* in such a way that the transformation is consistency preserving. In other words, if F has a model then *SCF* has a model as well. And if *SCF* is a contradiction then F is a contradiction as well. Thus the proof by general resolution is an indirect proof. The proof method is guarded by the following rules:

General *quantifier elimination*: $\forall x A(x) \vdash A(t/x)$, where the term t is substitutable for the variable x.

General *resolution rule*: Let A_i, B_i, l_1, l_2 be atomic formulas of FOL, σ a collision-less substitution such that $l_1\sigma = l_2\sigma$. Then the following rule is valid

$$\frac{A_1 \vee \ldots \vee A_m \vee l_1, B_1 \vee \ldots \vee B_n \vee \neg l_2}{A_1\sigma \vee \ldots \vee A_m\sigma \vee B_1\sigma \vee \ldots \vee B_n\sigma}$$

An *SCF* formula is not satisfiable if and only if the empty clause (#) is derivable from *SCF* by step-by-step application of the general resolution rule.

Method of Programming in Logic (Prolog) is a special case of the general resolution method with three major restrictions. First, it works only with *Horn clauses*, i.e. clauses with at most one positive literal:

$$HC = l \vee \neg m_1 \vee \ldots \vee \neg m_k.$$

Second, Prolog mostly applies a *goal driven*, *depth-first* strategy with *backtracking* for generating resolvents. Though this strategy is not complete, it is more effective than other strategies evaluating the computational tree completely. Third, *negation* is dealt with as a failure to prove (the Close World Assumption).

Notation in Prolog.
Rule (Conditional statement): P:- Q_1,...,Q_n.

Atomic formula $P = p(t_1,\ldots,t_m)$ is the *head* of the rule, t_i are its formal parameters; atomic formulas Q_1,...,Q_n is the *body* (*tail*) of the rule with the *subgoals* Q_i.

Note that conditional statement is equivalent to

$$(Q_1 \wedge \ldots \wedge Q_n) \supset P \text{ or i}(\neg Q_1 \vee \ldots \vee \neg Q_n \vee P).$$

Fact: P. A Horn clause without negative literals, i.e., an atomic formula.
Goals: ?- $Q_1, \ldots, Q_n$. A Horn clause without a positive literal: $\neg Q_1 \vee \ldots \vee \neg Q_n$.

Note that this formula is equivalent to $(Q_1 \wedge \ldots \wedge Q_n) \supset$ False.
An empty clause: #. (Contradiction, success).

Logic program is a sequence of rules and facts. Prolog inference machine derives answers to the questions (goals) by applying respective substitutions in order to instantiate variables in goals and generate resolvents. Derivation of an empty clause is a success; the respective substitution is then an answer to the goal.

3 Transition from TIL into Prolog

When encoding TIL constructions in Prolog, we focus on the constructions that construct empirical entities of type $o_{\tau\omega}$, $(o\alpha)_{\tau\omega}$ and $(o\alpha_1 \ldots \alpha_n)_{\tau\omega}$, i.e., propositions, properties and relations-in-intension, respectively, or mathematical facts of type o. Constructions of propositions are translated as Prolog facts/rules or *Yes–No questions* according whether a respective message is of a kind 'Inform' or 'Query'. Constructions of properties and relations are translated as *Wh-questions*.[5]

When creating a Prolog base of facts and rules we have to record the types of entities that receive mention in TIL constructions. In particular, types of intensions have to be distinguished from types of extensions. To this end we use a special Prolog predicate 'type'. Prolog base of facts and rules then contains Prolog translation of TIL constructions $v-$constructing True (or False in case of an incorrect data collection), i.e., values of the propositions in a given world w at a time t of data collection. When querying on this base, we use a construction of an intension and ask for its value in a state of affairs as recorded in or entailed by the Prolog base.

3.1 Building up a Prolog base of facts and rules

Prolog base of facts and rules is created by transferring indicative messages of the kind 'Inform', the content of which is encoded in the $\mathcal{TIL}$-`Script` language.

For instance, the proposition that Charles is a professor constructed by the Closure

$$\lambda w \lambda t [{}^0\mathit{Professor}_{wt}\ {}^0\mathit{Charles}]$$

and encoded in $\mathcal{TIL}$-`Script` by[6]

```
[\w:World \t:Time [['Professor@wt 'Charles]]].
```

is translated into a Prolog fact

```
prof(charles).                                    (1)
```

By means of the Prolog predicate 'type' we remember TIL type of *Professor*, i.e., the type $(o\iota)_\tau\omega$ of an individual property, in order to reconstruct the original proposition in the reverse translation into $\mathcal{TIL}$-`Script` .

The proposition that all professors are employees constructed by the Closure

$$\lambda w \lambda t\ \forall x [[{}^0\mathit{Professor}_{wt}\ x] \supset [{}^0\mathit{Employee}_{wt}\ x]]$$

is translated into the Prolog rule (accompanied by the respective type predicates)

```
empl(X):- prof(X).                                (2)
```

[5] For details on the analysis of questions and answers in TIL see, e.g., [5]. [6] For details on the $\mathcal{TIL}$-`Script` language see, e.g., [1] or [2].

Similarly the proposition that cars and bicycles are mobile agents constructed by

$$\lambda w \lambda t\ \forall x[[[^0Car_{wt}\ x] \vee [^0Bike_{wt}\ x]] \supset [[^0Mobile\ ^0Agent]_{wt}\ x]]$$

is translated into the Prolog rule (accompanied by the respective type predicates)

```
mobile_agent(X):- car(X); bike(X).
```
(3)

So far so good. However, TIL is based on the functional approach whereas Prolog is a relational language. Thus the translation of constructions containing constituents $v-$constructing functions of types $(\alpha\beta)$ and attributes (empirical functions) of types $(\alpha\beta)_{\tau\omega}$, where $\alpha \neq o$, is not so straightforward. We have to first transform them into constructions of relations of type $(o\alpha\beta)_{\tau\omega}$ by introducing an auxiliary variable.

For instance, the fact that Charles' only car is a mobile agent is constructed by

$$\lambda w \lambda t[[^0Mobile\ ^0Agent]_{wt}\ [^0The[^0Car\_of_{wt}\ ^0Charles]]]$$

Types. $Mobile/((o\iota)_{\tau\omega}(o\iota)_{\tau\omega})$ – modifier; $Agent/(o\iota)_{\tau\omega}$; $Car\_of((o\iota)\iota)_{\tau\omega}$; $The/(\iota(o\iota))$ – the singulariser ('the only ...').
The respective code in the $\mathcal{T}IL$-`Script` language is this:

```
[\w:World [\t:Time [
        ['Mobile 'Agent]@wt ['Sing ['Car_of@wt 'Charles]]
]]].
```

We have to simplify this construction by ignoring the singulariser, and transform this Closure into the Prolog rule

```
mobile_agent(Y):- car_of(Y,charles).
```

Gloss. For all y, if y is a car of Charles then y is a mobile agent.

In general, the pre-processing of constructions of empirical functions in TIL is driven by this strategy. If $Attr \rightarrow (\alpha\beta)_{\tau\omega}$ is a construction of an empirical function, $P \rightarrow (o\alpha)_{\tau\omega}$ a construction of an α-property and b an entity of type β, then the construction of a proposition of the form

$$\lambda w \lambda t[P_{wt}\ [Attr_{wt}\ ^0b]]$$

is modified into $(x \rightarrow \alpha)$

$$\lambda w \lambda t[\forall x[[x = [Attr_{wt}\ ^0b]] \supset [P_{wt}\ x]]]$$

which is then translated into the Prolog rule

```
p(X) :- attr(X,b).
```

Moreover, TIL functional overlay over the so-created Prolog base of facts and rules must also make it possible to distinguish between analytically necessary facts and rules and empirical ones.

For instance, the above proposition that cars and bicycles are mobile agents could be specified as an analytical one, meaning that *necessarily*, all cars or bicycles are mobile agents. The respective TIL analysis is then this Closure

$$\forall w \forall t \forall x [[[^0Car_{wt}\, x] \vee [^0Bike_{wt}\, x]] \supset [[^0Mobile\, {}^0Agent]_{wt}\, x]].$$

However, the resulting Prolog rule will be the same as the above rule (3). In such a case we must remember in the TIL ontology that the property of being a mobile agent is a *requisite* of the properties of being a car and being a bicycle.

Another problem we meet when translating TIL constructions into Prolog is the problem of *equivalences.* For instance, inserting a formula of the form $(p \equiv q) \Leftrightarrow (p \supset q) \wedge (q \supset p)$ into the Prolog base might cause the inference machine to be kept in an infinite loop. Yet we need to record such equivalences in TIL ontology. This is in particular the case of a meaning refined by *definition.*

Imagine, for instance, the situation of an agent *a* who does not have in its ontology the concept of a car park with vacancies. Then *a* may learn by asking the other agents that "A car park with vacancies is a car park some of whose parking spaces nobody has occupied yet". The content of a 'query' message asking for the definition of 'car park with vacancies' is [^{0}*Unrecognized* ^{0}[^{0}*Vac* ^{0}*Car_Park*]]. The reply message content is

[^{0}*Refine* ^{0}[^{0}*Vac* ^{0}*Car_Park*]
^{0}[$\lambda w \lambda t \lambda x [[^0Car\_Park_{wt}\, x] \wedge \exists y [[^0Space\_of_{wt}\, y\, x] \wedge \neg [^0Occupied_{wt}\, y]]]]]$.

Thus the constructions [^{0}*Vac* ^{0}*Car_Park*] and

$$[\lambda w \lambda t \lambda x [[^0Car\_Park_{wt} x] \wedge \exists y [[^0Space\_of_{wt}\, y x] \wedge \neg [^0Occupied_{wt}\, y]]]]$$

are *ex definitione* equivalent by constructing one and the same property. This fact can be encoded in Prolog only by a general rule:

```
vac_park(X,Y) :- park(X), space_of(Y,X), non_occupied(Y).
```

Gloss. (For all X, Y), X has a parking vacancy Y if X is a car park and Y is a space of X and Y is not occupied.

3.2 Querying in Prolog

As stated above, messages of the kind 'Query' with the semantic content constructing propositions represent *Yes–No questions* on the truth-value of the respective proposition in a given state-of-affairs. The semantic core of a Yes–No question (the content of the respective message) is thus the same as that of a corresponding indicative sentence; a construction of a proposition.

For instance, the above example of the proposition that "Charles is a professor" is encoded by a message with the same content as the corresponding query "Is Charles a professor?". The two messages differ only in their kind. The former is of the kind 'Inform', the latter of the kind 'Query'.

However, the Prolog code will differ. We now *ask* whether this proposition is entailed by the recorded Prolog knowledge base. To this end we must translate

the message as a *negative* Prolog fact so that Prolog inference machine can check whether the negated proposition contradicts the base:

```
?- prof(charles).
```

In this case Prolog answers `Yes` providing the fact (1) is contained in its base.

We can also put Yes–No questions on the existence of individuals with such and such properties. In such a case Prolog would answer not only simple `Yes`, but also *which* individuals (according to its knowledge) are such and such.

For instance, the query "Are some employees professors?" is analysed by TIL Closure ($Employee$, $Professor/(o\iota)_{\tau\omega}$; $x \rightarrow \iota$)

$$\lambda w \lambda t [\exists x [[^0 Employee_{wt}\, x] \wedge [^0 Professor_{wt}\, x]]].$$

The $\mathcal{T}IL$-`Script` encoding is then

```
[\w:World [\t:Time [Exists x:Indiv [
        'And ['Empl@wt x] ['Prof@wt x]
]]]].
```

And the respective Prolog code is this

```
?- empl(X), prof(X).
```

Now Prolog tries to succeed in deriving these two goals. To this end Prolog searches its knowledge base for facts and rules containing the `empl` or `prof` in their head, and if possible Prolog unifies the variable `X` with the respective arguments of the found rules/facts. In our case `empl(X)` first matches with (2) and then `prof(X)` with (1) resulting in the success by substituting `charles` for `X`. Thus Prolog answer is `Yes, X=charles`.

When raising '*wh-questions*' we ask for the value of an α-property or an α-relation-in-intension in a given world w at time t. As explained in [5], the analysis is driven by a possible answer. If an answer is a set of individuals, we construct a property. In general, if an answer is a set of tuples of $\alpha_1, \ldots, \alpha_m$-objects, we construct a relation-in-intension of type $(o\alpha_1 \ldots \alpha_m)_{\tau\omega}$.

For instance, the answer to the question "Which professors are older than 60 years?" can be {Materna, Duží}, i.e., the set of individuals. Thus we have to construct the property of individuals:

$$\lambda w \lambda t \lambda x [[^0 Professor_{wt}\, x] \wedge [^0 > [^0 Age_{wt}\, x]\, {}^0 60]],$$

where Age is an attribute of type $(\tau\iota)_{\tau\omega}$. Again, the respective $\mathcal{T}IL$-`Script` code is

```
[\w:World [\t:Time [ \x:Indiv [
        'And ['Prof@wt x] ['> ['Age@wt x] '60]
]]]].
```

In order to translate this question into Prolog, we have to convert the attribute *Age* into a binary relation using an auxiliary variable in a similar way as described above. The resulting construction is

$$\lambda w \lambda t \lambda x [\exists y [[{}^0Professor_{wt}\, x] \wedge [[y = [{}^0Age_{wt}\, x]] \wedge [{}^0> \; y \; {}^0 60]]]].$$

This Closure is transformed into three Prolog goals as follows:

```
?- prof(X), age(Y,X), Y>60.
```
[7]

Now there is another problem generated by Prolog relational approach. Whereas a functional program would return the value of the above property recorded in a given knowledge base state, i.e., a *set* of individuals, Prolog answer in case of the success in meeting these goals will be, e.g., `Yes, X=Duzi, Y=60`. If we want another individual instantiating the property, we have to keep entering semicolon, `';'`, until obtaining the answer `No`.

In order to overcome this difficulty, we may use Prolog '*findall*/3' predicate which has three arguments: (*what, from where, to where*). However, in such a case we have to specify *one* goal (*from where*). Here is how. We create a new unique name of the property the instances of which are asked for. For instance, the above property can be coined 'Aged_professor', and the equality definition "Aged professors are professors older than 60 years" inserted as follows:

$$\lambda w \lambda t \lambda x [{}^0Aged\_professor_{wt} x] = \lambda w \lambda t \lambda x [[{}^0Professor_{wt} x] \wedge [{}^0> [{}^0Age_{wt}\, x] \, {}^0 60]].$$

However, as explained above, in Prolog we cannot deal with equivalences, because equivalence would cause Prolog attempting to succeed in meeting a goal *ad infinitum.* Thus we have to insert a new *rule* the head of which is the *definiendum* and tail is the *definiens* of the inserted definition. The resulting Prolog code is

```
?-assert(aged_prof(X):-prof(X), age(Y,X), Y>60.),
    findall(X,aged_prof(X),Result),
    retract(aged_prof(X):-prof(X), age(Y,X), Y>60.).
```
[8]

The predicate `assert` serves for inserting the auxiliary rule into Prolog base, and `retract` for erasing it.

The process of generating a generic property corresponding to a 'wh--question' is this:

1) Transform the TIL compound construction of a property into an equivalent construction by inserting a new unique primitive concept of the property. If $C^1, \ldots, C^m$ are the respective constituents of the compound Closure, then the schema of TIL pre-processing is as follows:
   $\lambda w \lambda t \lambda x [{}^0C\_new_{wt}\, x] = \lambda w \lambda t \lambda x [[C^1_{wt} x] \wedge \ldots \wedge [C^m_{wt}\, x]]$.
2) Translate into Prolog: `c_new(X):-c1(X),...,ck(X).`

[7] Due to Prolog depth-first, left-to-right evaluation strategy the order of the goals is important here. The variable `Y` has to be instantiated first with the value of `age_of X`, and then compared with `60`.

[8] The method of processing the attribute *Age_of* is similar as above and it is described in Section 3.3.1.

3) If the rule 'cnew' is contained in Prolog base, then
`findall(X, c_new(X), Result).`
Otherwise, i.e., if 'cnew' is not contained in Prolog base, then
```
?-not(c_new(X)), assert(c_new(X):-c1(X), ..., ck(X)),
    findall(X, c_new(X), Result),
    retract(c_new(X):-c1(X), ..., ck(X)).
```

3.3 Schema of the transition from TIL into Prolog

It should be clear by now that prior to the translation of TIL constructions into Prolog, a lot of pre-processing has to be done in TIL. These modifications vary according to the structure of constructions and the type of entities constructed by primitive concepts, i.e., Trivialisations of non-constructive entities. These primitive concepts fully determine a conceptual system within which we work, and they have to be contained in the respective ontology together with their types. The choice of a conceptual system depends, of course, on the domain area and also on application goals and problems to be solved.

By way of summary we now present a general schema for TIL pre-processing and translation of particular constructions into Prolog code according to the type of entities that receive mention by constituents of a construction. Recall that types of the entities have to be also remembered by Prolog, which we do not indicate here.

Questions are expressed as constructions of intensions, the extension of which in a particular world w and time t we want to know. In case of wh-question the respective constructed entity is a property or relation. In case of Yes–No question it is a proposition. In case of mathematical questions we consider constructions of mathematical functions the value of which we want to compute. In what follows we use double arrow '$\Rightarrow$' for 'translated or pre-processed into'.

3.3.1 Analysis and translation of simple facts or *wh-queries*. First we present a schema of pre-processing simple constructions that do not contain constituents of truth-value functions and quantifiers. Let constructions *P, Attr, Rel* and *Calc* be constructions of such a simple form.

1. Constitents of *propositions*. Let $P \rightarrow o$, Then $\lambda w \lambda t P \Rightarrow$ `?-p.` or `p.`
   Example. "Is Charles a professor?";
   $\lambda w \lambda t\, [{}^0Prof_{wt}\ {}^0Charles] \Rightarrow$ `?- prof(charles).`
2. Constituents of *α-properties*. Let $P \rightarrow (o\alpha)_{\tau\omega}$; $x \rightarrow \alpha$. Then
   $\lambda w \lambda t \lambda x [P_{wt} x] \Rightarrow$ `?-p(X).`
   Example. "Who are our the associate professors?";
   $\lambda w \lambda t \lambda x [[{}^0Associate\ {}^0Prof]_{wt}\ x] \Rightarrow$ `?-assoc_prof(X).`
3. Constituents of $(\alpha\beta)$-*attributes*. Let $Attr \rightarrow (\alpha\beta)_{\tau\omega}$, $P \rightarrow (o\alpha)_{\tau\omega}$; b/β; $\alpha \neq o$; $x \rightarrow \beta$; $y \rightarrow \alpha$.
   Now we have to distinguish two cases.

a) The constituent is used in an 'Inform' message, the content of which is of the form $\lambda w\lambda t\,[P_{wt}[Attr_{wt}\,b]]$. Then
$\lambda w\lambda t\,[P_{wt}\,[Attr_{wt}\,b]] \Rightarrow \lambda w\lambda t\,[\forall y[[^0 = y\,[Attr_{wt}\,b]] \supset [P_{wt}\,y]]]$
$\Rightarrow$ `p(Y) :- attr(Y,b).`
Example. "Charles' father is a professor".
$\lambda w\lambda t\,[^0Professor_{wt}[^0Father\_of_{wt}\,{}^0Charles]] \Rightarrow$
$\lambda w\lambda t\,[\forall y[[^0 = y\,[^0Father_{wt}\,{}^0Charles]] \supset [^0Professor_{wt}\,y]]]$
$\Rightarrow$ `prof(Y) :- father(Y,charles).`

b) The constituent is used in a 'Query message' and it is of the form $\lambda w\lambda t\lambda x[P_{wt}[Attr_{wt}\,x]]$. Then
$\lambda w\lambda t\lambda x[P_{wt}[Attr_{wt}\,x]] \Rightarrow \lambda w\lambda t\lambda x[\exists y[[^0 = y[Attr_{wt}\,x]] \wedge [P_{wt}\,y]]]$
$\Rightarrow$ `?- attr(Y,X), p(Y).`
Example. "Whose father is a professor?".
$\lambda w\lambda t\lambda x[^0Professor_{wt}[^0Father\_of_{wt}\,x]] \Rightarrow$
$\lambda w\lambda t\lambda x[\exists y[[^0 = y[^0Father_{wt}\,x]] \wedge [^0Professor_{wt}\,y]]]]$
$\Rightarrow$ `:- father(Y,X), prof(Y).`

4. Constituents of *relations-in-intension*. Let $Rel \rightarrow (o\alpha\beta)_{\tau\omega}$; $x \rightarrow \alpha$; $y \rightarrow \beta$. Then
$\lambda w\lambda t\lambda x\lambda y[Rel_{wt}\,xy] \Rightarrow$ `?-rel(X,Y).`
Example. "Who is affiliated to whom?";
$\lambda w\lambda t\lambda x\lambda y[^0Affiliate_{wt}xy] \Rightarrow$ `?-affiliate(X,Y).`
5. Constituents of *mathematical functions*. Let $Calc \rightarrow (\tau\tau)$; $x, y \rightarrow \tau$; m/τ. Then
$[Calc\ {}^0m] \Rightarrow$ `?- Y is calc(m).`
$\lambda x\,[Calc\,x] \Rightarrow \lambda x\lambda y[^0 = y[Calc\,x]] \Rightarrow$ `?-Y is calc(X).`
In general, constituents of $n-$ary mathematical functions are transferred as follows. Let $Calc^n/(\tau\tau\ldots\tau)$. Then
$\lambda x_1\ldots x_n[Calc\,x_1\ldots x_n] \Rightarrow \lambda x_1\ldots x_n\lambda y[^0 = y[Calc\,x_1\ldots x_n]] \Rightarrow$
`?- Y is calc(X1,...,Xn).`
Example. "Which is the value of the square root of 144?";
$[^0Square\_root\ {}^0144] \Rightarrow$ `?- Y is square_root(144).`

3.3.2 Analysis and translation of more complex *wh-queries*. Since Prolog queries can be only Horn clauses without a positive literal, i.e., clauses of a form $\neg l_1 \vee \neg l_2 \vee \ldots\ \neg l_m$, their general form is $\neg(l_1 \wedge l_2 \wedge \ldots\ l_m)$, which in Prolog notation is recorded as ?- l_1, l_2, …,l_m. The clauses l_1,…,l_m are the goals to be met.

Thus a TIL construction to be translated must be a conjunctive construction of a property or relation, or of an existential proposition. Let $C_1, \ldots, C_m$ be simple constructions not containing constituents of truth-value functions and quantifiers. Then the general form of TIL construction translatable into a Prolog *wh*-question is:

$$\lambda w\lambda t\lambda x_1\ldots x_n[C_1 \wedge \ldots \wedge C_m], \text{ or } \lambda w\lambda t[\exists x_1\ldots x_n[C_1 \wedge \ldots \wedge C_m]].$$

Schema of the resulting Prolog wh-query is

```
?- c1, ..., ck.
```

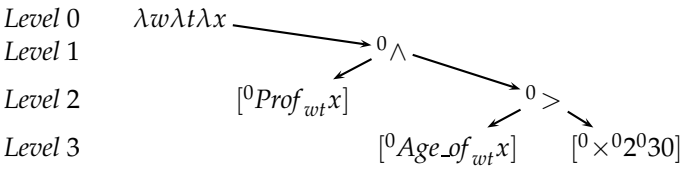
where the `c1, ..., ck` are simple goals created according to the items (1)–(5) of the proceeding paragraph.

According to the complexity of a construction C we first create the structure tree of C, leafs of which are simple constituents of C. The pre-processing and translation of C is then executed bottom up, from leafs to the root of the tree.

Example. "Who are the professors older than 2×30 years?".
The analysis is a construction of the property of being a professor older than 2×30 years:

$$\lambda w \lambda t \lambda x [{}^0\wedge [{}^0Prof_{wt}\, x][{}^0> [{}^0Age\_of_{wt}\, x][{}^0\times\, {}^02\, {}^030]]].$$

The structure tree is this:



Beginning with the bottommost level 3, we transform leafs containing attributes and mathematical functions in compliance with the above rules (3) and (5), respectively, into constituents $[{}^0 = y\ [{}^0Age\_of_{wt}\, x]]$ and $[{}^0 = z[{}^0\times {}^02\ {}^030]]$. The auxiliary variables y, z become the arguments of the parent node at level 2: $[{}^0 > y\, z]$. The node $[{}^0Prof_{wt}\, x]$ remains without change, as well as the parent node ${}^0\wedge$ of level 1, because they $v-$construct a truth-value of type o. Finally we have to adjust the root level 0 in compliance with (3) above into $\lambda w \lambda t \lambda x \lambda y \lambda z$ ... The resulting construction is

$$\lambda w \lambda t \lambda x \lambda y \lambda z [{}^0\wedge [{}^0Prof_{wt}\, x][{}^0\wedge [{}^0= y[{}^0Age\_of_{wt}\, x]][{}^0\wedge [{}^0= z[{}^0\times\, {}^02\, {}^030]][{}^0>\, y\, z]]]].$$

The respective Prolog code is then

```
?- prof(X),age_of(Y,X), Z is ×(2,30), Y>Z.
```

As stated above, if we want *all* the professors older than 2×30 years, we have to insert a new unique name of the property by the equation definition
$\lambda w \lambda t \lambda x [{}^0Aged\_professor_{wt} x] =$
$\lambda w \lambda t \lambda x [\exists y \exists z [{}^0\wedge [{}^0Prof_{wt}\, x]$
$[{}^0\wedge [{}^0= y[{}^0Age\_of_{wt}\, x]][{}^0\wedge [{}^0= z[{}^0\times {}^02{}^030]][{}^0> yz]]]]].$
The resulting Prolog code is

```
?-assert(aged_prof(X):-prof(X),age_of(Y,X),Z is ×(2,30),Y>Z.),
findall(X,aged_prof(X),Result).
```

3.4 Specification of a TIL subset transferable into Prolog code

Up to now we considered only constructions $v-$constructing entities of a type $(o\beta)$, the structure tree of which is conjunctive. Using a terminology of predicate logic, we considered only constructions which are in a generalised

conjunctive normal form $\lambda w \lambda t\ \lambda x_1 \dots x_m\ [C_1 \wedge \dots \wedge C_m]$, where $C_1, \dots, C_m$ are simple constructions not containing constituents of truth-value functions and quantifiers.

In order to specify the subset of TIL that can be encoded in Prolog, we have to specify restrictions on TIL propositional constructions implied by Prolog, and describe an algorithm of transferring a closed construction $C \rightarrow o_{\tau\omega}$ into the Skolem clausal form (SCF). Here is how.

In what follows we now use '$C(x,y,\dots)$' as a schema of a construction containing free variables x, y, ...

1. *Eliminate w,t.* $[\lambda w \lambda t C(w,t)] \Rightarrow C$
2. *Eliminate unnecessary quantifiers* that do not quantify any variable.
   (For instance, Composition of the form $[[\exists y \forall z P(z)] \wedge [\forall x \exists v Q(v)]]$ is adjusted into $[[\forall z P(z)] \wedge [\exists v Q(v)]]$.)
3. *Apply α-rule* in such a way that different λ-bound variables have different names.
   (For instance, Composition $[[\forall x P(x)] \wedge [\forall x Q(x)]]$ is adjusted into the Composition $[[\forall x P(x)] \wedge [\forall y Q(y)]]$.)
4. *Eliminate connectives* $\supset$ and $\equiv$ by applying the rules $[C \supset D] \vdash [\neg C \vee D]$ and $[C \equiv D] \vdash [[\neg C \vee D] \wedge [\neg D \vee C]]$.
5. *Apply de Morgan laws*:
   $\neg[C \wedge D] \vdash [\neg C \vee \neg D]$, $\neg[C \vee D] \vdash [\neg C \wedge \neg D]$,
   $\neg[\exists x \neg C(x)] \vdash [\forall x C(x)]$, $\neg[\forall x \neg C(x)] \vdash [\exists x C(x)]$.
6. If the resulting construction C contains now existential quantifiers, then *reject* C as non-transferable.
   (For instance, Composition of the form $[[\exists y P(y)] \wedge [\forall x \exists v Q(x,v)]]$ is not transferable into Prolog.)
7. *Move general quantifiers to the left.*
   (For instance, Composition $[[\forall x P(x)] \wedge [\forall y Q(y)]]$ is adjusted into the Composition $[\forall x[\forall y\ [P(x) \wedge Q(y)]]]$.)
8. *Apply distributive laws*:
   $[[C \wedge D] \vee E] \vdash [[C \vee E] \wedge [D \vee E]]$, $[C \vee [D \wedge E]] \vdash [[C \vee D] \wedge [C \vee E]]$.

The resulting construction is now of the form

$$\forall x_1 [\forall x_2 \dots [\forall x_n [C_1 \wedge \dots \wedge C_m]]]],$$

where C_i are clauses, i.e., disjunctions of simple 'positive/negative' constructions (not containing constituents of truth-value functions $\wedge$, $\vee$, $\supset$, $\equiv$ and quantifiers). If for some i the clause C_i contains more than one positive disjunct, then *reject* the construction as not transferable into Prolog code. Otherwise, insert the translation of $C_1, \dots, C_m$ into Prolog program base.

Example. Consider the base of sentences.

- "All mobile agents are cars or bicycles".
- "If Charles is driving on highway D1 then he is a mobile agent".
- "Paul is not a mobile agent or it is not true that if he lives in New York he drives his car".

– "It is not true that some agents are neither mobile nor infrastructure agents".

a) Type-theoretical analysis.
Mobile,Infra(structure)$/((o\iota)_{\tau\omega}(o\iota)_{\tau\omega})$; *Agent*, *Car*, *Bike*, *High_Way*, *Employee*$/(o\iota)_{\tau\omega}$; *Live_in*, *Car_of*$/(o\iota\iota)_{\tau\omega}$; *Charles*, *Paul*, *D*1, *NY*$/\iota$.
b) Synthesis and pre-processing.

First sentence. $\lambda w\lambda t\ [\forall x[[{}^0\!Mobile\ {}^0\!Agent]_{wt}\ x] \supset [[{}^0\!Car_{wt}\ x] \vee [{}^0\!Bike_{wt}\ x]]]]$
$\Rightarrow$ (Step 1) $[\forall x[[{}^0\!Mobile\ {}^0\!Agent]x] \supset [[{}^0\!Car\,x] \vee [{}^0\!Bike\,x]]]]$
$\Rightarrow$ (Step 4) $[\forall x \neg[[{}^0\!Mobile\ {}^0\!Agent]x] \vee [{}^0\!Car\,x] \vee [{}^0\!Bike\,x]]]$

This construction is not transferable into Prolog code, because it contains two positive constituents, namely $[{}^0\!Car\,x]$ and $[{}^0\!Bike\,x]$.

Second sentence.
$\lambda w\lambda t\ [[{}^0\!Drive_{wt}\ {}^0\!Charles\ {}^0\!D1] \wedge [{}^0\!High\_Way_{wt}\ {}^0\!D1]]] \supset [[{}^0\!Mobile\ Agent]_{wt}\ {}^0\!Charles]]]$
$\Rightarrow$ (1) $[[{}^0\!Drive\ {}^0\!Charles\ {}^0\!D1] \wedge [{}^0\!High\_Way\ {}^0\!D1]]] \supset [[{}^0\!Mobile\ Agent]\ {}^0\!Charles]]]$
$\Rightarrow$ (4) $[\neg[{}^0\!Drive\ {}^0\!Charles\ {}^0\!D1] \vee \neg[{}^0\!High\_Way\ {}^0\!D1]] \vee [[{}^0\!Mobile\ Agent]\ {}^0\!Charles]]]$.
The construction is transferable into a Prolog rule:

```
mobile_agent(charles):-drive(charles,d1),high_way(d1).
```

Third sentence.
$\lambda w\lambda t[\neg[[{}^0\!Mobile\ {}^0\!Agent]_{wt}\ {}^0\!Paul] \vee$
$\neg[[{}^0\!Live\_in_{wt}\ {}^0\!Paul\ {}^0\!NY] \supset [{}^0\!Drive_{wt}\ {}^0\!Paul\ [{}^0\!Car\_of_{wt}\ {}^0\!Paul]]]$
$\Rightarrow$ (1) $[\neg[[{}^0\!Mobile\ {}^0\!Agent]\ {}^0\!Paul] \vee$
$\neg[[{}^0\!Live\_in\ {}^0\!Paul\ {}^0\!NY] \supset [{}^0\!Drive\ {}^0\!Paul\ [{}^0\!Car\_of\ {}^0\!Paul]]]$
$\Rightarrow$ (4,5) $[\neg[[{}^0\!Mobile\ {}^0\!Agent]\ {}^0\!Paul] \vee$
$[[{}^0\!Live\_in\ {}^0\!Paul\ {}^0\!NY] \wedge \neg[{}^0\!Drive\ {}^0\!Paul\ [{}^0\!Car\_of\ {}^0\!Paul]]]$
$\Rightarrow$ (8) $[\neg[[{}^0\!Mobile\ {}^0\!Agent]\ {}^0\!Paul] \vee [{}^0\!Live\_in\ {}^0\!Paul\ {}^0\!NY]] \wedge$
$[\neg[[{}^0\!Mobile\ {}^0\!Agent]\ {}^0\!Paul] \vee \neg[{}^0\!Drive\ {}^0\!Paul\ [{}^0\!Car\_of\ {}^0\!Paul]]]]$
The construction is transferable into a Prolog rule and three goals:

```
Live_in(paul,ny):-mobile_agent(paul).
:-mobile_agent(paul), drive(paul,Y), car_of(Y,paul).
```
[9]

Fourth sentence.
$\lambda w\lambda t\neg[\exists x[[{}^0\!Agent_{wt}\ x] \wedge \neg[[{}^0\!Mobile\ {}^0\!Agent]_{wt}x] \wedge \neg[[{}^0\!Infra\ {}^0\!Agent]_{wt}\ x]]]$
$\Rightarrow$ (1) $\neg[\exists x\ [[{}^0\!Agent\,x] \wedge \neg[[{}^0\!Mobile\ {}^0\!Agent]x] \wedge \neg[[{}^0\!Infra\ {}^0\!Agent]x]]]$
$\Rightarrow$ (5) $[\forall x\ [\neg[{}^0\!Agent\,x] \vee [[{}^0\!Mobile\ {}^0\!Agent]x] \vee [[{}^0\!Infra\ {}^0\!Agent]x]]]$
The construction is not transferable into a Prolog rule.

Remark. The first step of pre-processing a construction consists in elimination of variables w, t. The result is an improper construction due to wrong typing. For instance, the Composition $[{}^0\!Agent\,x]$ is not well formed, because the property *Agent*$/(o\iota)_{\tau\omega}$ has to be extensionalised first, and only then applied to an individual. Yet, since together with the resulting Prolog rules, facts and goals we remember TIL types, the reverse translation into TIL will be correct.

[9] Since *Car_of* is an attribute, the Composition $[{}^0\!Drive\ {}^0\!Paul\ [{}^0\!Car\_of\ {}^0\!Paul]]$ is processed by means of the auxiliary variable y; see Section 3.3.1.

4 Conclusion

We have introduced a method of building up an interface between the functional $\mathcal{T}IL$-`Script` language and relational Prolog language. By the transition of TIL into Prolog we gain the inference machine of Prolog. The value we have to pay is rather high. We have to build a powerful TIL functional overlay in order not to lose information, and to modify TIL constructions into the form that can be processed by Prolog. Of course, due to the high expressive power of TIL, only a subset of TIL constructions is transferable. Thus we also specified an algorithm that decides which constructions are transferable, and as a result it produces an adjusted construction specified in the Prolog code.

The direction for future research is clear. We have to extend the method to involve partiality and hyper-intensional features of TIL in its full power. To this end the restrictions applied by Prolog seem to be too tight. Thus we will extend the method into an implementation of the full TIL inference machine. Yet the clarity of this direction does not imply its triviality. The complexity of the work going into building such an inference machine is almost certain to guarantee that complications we are currently unaware of will crop up. A sensible approach will be to develop the inference machine by involving the methods of functional programming and extend them to involve partiality and hyper-intensional, i.e., procedural features.

Acknowledgements. This research has been supported by the program 'Information Society' of the Czech Academy of Sciences within the project "Logic and Artificial Intelligence for multi-agent systems", No. 1ET101940420.

References

1. Bratko, I.: *Prolog Programming for Artificial Intelligence.* Addison-Wesley, 2001.
2. Ciprich, N., Duží, M., Košinár, M.: $\mathcal{T}IL$-`Script` : Functional Programming Based on Transparent Intensional Logic. In: *RASLAN 2007*, Sojka, P., Horák, A., (Eds.), Masaryk University Brno, 2007, pp. 37–42.
3. Ciprich, N., Duží, M. and Košinár, M.: *The $\mathcal{T}IL$-`Script` language*. To appear in the Proceedings i of the 18th European Japanese Conference on Information Modelling and Knowledge Bases (EJC 2008), Tsukuba, Japan 2008.
4. Duží, M. Ďuráková, D., Děrgel, P., Gajdoš, P., Müller, J.: Logic and Artificial Intelligence for multi-agent systems. In: Marie Duží, Hannu Jaakkola, Yasushi Kiyoki and Hannu Kangassalo (editors), *Information Modelling and Knowledge Bases* XVIII, Amsterdam: IOS Press, pp. 236–244.
5. Duží, M. and Materna, P. Reprezentace znalostí, analýza tázacích vět a specifikace dotazů nad konceptuálním schématem HIT. In: Dušan Chlápek (Ed.), *Datakon 2002*, Brno, pp. 195–208.
6. Foundation for Intelligent Physical Agents. `http://www.fipa.org/`.
7. Tichý, P. *The Foundations of Frege's Logic*. Walter de Gruyter, Berlin-New York, 1988.
8. Tichý, P. (2004): *Collected Papers in Logic and Philosophy*, V. Svoboda, B. Jespersen, C. Cheyne (Eds.), Prague: Filosofia, Czech Academy of Sciences, and Dunedin: University of Otago Press.

TIL in Knowledge-Based Multi-Agent Systems

Tomáš Frydrych, Ondřej Kohut, Michal Košinár

VŠB–Technical University Ostrava
17. listopadu 15, 708 33 Ostrava, Czech Republic
tomas.frydrych.st@vsb.cz, ondrej.kohut.fei@vsb.cz, michal.kosinar@vsb.cz

Abstract. Transparent Intensional Logic (TIL) is a highly expressive logic system. Its potential applications in artificial intelligence and multi-agent systems are broad. We introduce the TIL-Script language, which is a computational variant of the language of TIL constructions. TIL-Script can be used as an alternative to FIPA SL language, whenever communication in a natural language is in need (e. g. human / computer interaction) and/or another systems are insufficient in their expressive power (for example in the area of knowledge representation for resource bounded knowledge based agents).

Key words: TIL, TIL-Script, Transparent Intensional Logic, Multiagent Systems, content language, FIPA

1 Introduction

Multi-agent systems are a relatively new technology which is still rapidly developing. One of the main problems a multi-agent system must deal with is communication and reasoning of agents. Current content languages are based on the first-order mathematical logic paradigm, extending the first-order framework whenever needed. However, these extensions are mostly specified syntactically, and their semantics is not very clear.

We propose the TIL-Script language based on Transparent Intensional Logic (TIL) which is a general (philosophical) logic system primarily designed for the logical analysis of natural languages. TIL-Script is well suited for the use as a content language of agent messages and it has a rigorously defined semantics. It can be also used as a general semantic framework for the specification and semantic analysis of other formal languages.

The paper is organized as follows. Section 2 describes the basic principles applied in agents' communication and interaction with environment. In Section 3 we discuss and critically examine FIPA SL content language. In Section 4 we briefly introduce the basic notions of TIL. In Section 5 ontologies and knowledge base model for TIL Script are briefly described. Finally, an example of agents' communication is presented in Section 6, and concluding remarks in Section 7.

Petr Sojka, Aleš Horák (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2008, pp. 31–40, 2008.

2 Multi-Agent Systems and Communication

Technologies based on agents are relatively new and very promising. Numerous applications of multi-agent technology can be found in artificial intelligence and large computer systems. A road-map of this approach is presented in [5]. In this paper we do not intend to deal with multi-agent systems (MAS) in general. Instead, we focus on communication in MAS and particularly on content languages.

Basic standards for MAS are given by FIPA (The Foundation for Intelligent Physical Agents, see [4,3]). According to it basic unit of communication is a message. It can be of an arbitrary form, but it is supposed to have a structure containing several attributes. *Content* of a message is one of these attributes.

From the point of view of communication logic the most important attributes are:

Performative denotes a type of the message – its communicative act. Basic performatives are: *Query, Inform and Request*.

Content carries the semantic of the message. It can be encoded in any suitable language.

Ontology is a vocabulary of domain specific terms. These (and only these) terms can be used in the content of the message.

2.1 Agent and Environment Interaction

In order to introduce communication based on agents' knowledge, we are going to describe agents' reactions to the events in their environment, and interaction with the environment in general. Figure 1 illustrates agents' interaction with the environment.

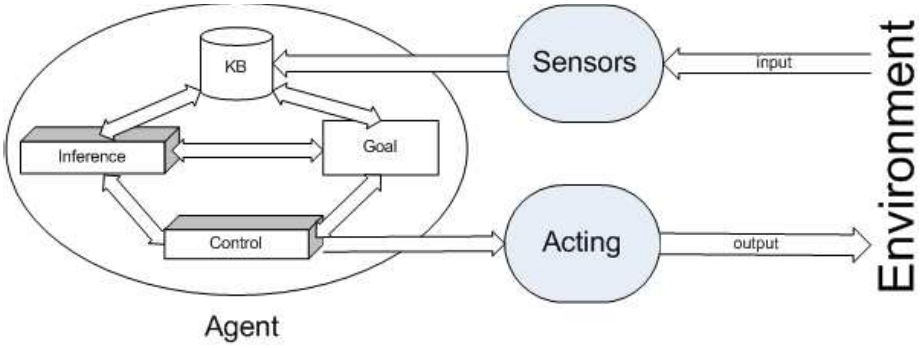


Fig. 1. Behaviour of agents in a real environment

Agents are autonomous, rational and goal-oriented. In order they can actively react on the events in their environment, they have to be equipped with:

- *Sensors* – "Ears", "Eyes"
- *Acting parts* – "Mouth" for communication, "Limbs" for an active reaction (movement etc.)
- *Knowledge-Base* based on ontologies. This part serves as an agents' memory that makes it possible to store perceived or learnt facts, entailed knowledge as well as general rules. (At least a minimal) ontology is needed to be shared with other agents, so that the agents' understand each other.
- *Inference engine* that is based on the TIL-Script language (or Description Logic, FIPA SL, etc.)
- *Goals* are the purpose of agents' life. An agent attempts to meet the goal assigned to them by applying their explicit knowledge stored in the knowledge base, and or inferred by the inference machine.
- *Control* part executes the actions to in accordance with a given agent's goal. In this way the agent influence its environment.

3 FIPA SL

One of the objectives of this paper is to propose a new content language for agents' communication in multi-agent systems. The languages like FIPA SL (Semantic Language) and KIF are mostly based on the First-Order Logic (FOL) paradigm, enriched with higher-order constructs wherever needed.[1] The enrichments extending FOL are well defined syntactically, while their semantics is often rather sketchy, which may lead to communication inconsistencies. Moreover, the bottom-up development from FOL to more complicated cases yields the versions that do not fully meet the needs of the MAS communication. In particular, agents' attitudes and anaphora processing create a problem. In the paper we are going to demonstrate the need for an expressive logical tool of Transparent Intensional Logic (TIL) in order to encode the semantic content of messages in a near-to-natural language. Using TIL, the human-computer interface and communication is designed in a smooth way.

In this section we now briefly discuss the existing standard, FIPA SL, which is the only FIPA candidate content language for a 'standard'.

One of advantages of the FOL approach is that FOL is a broadly used and well elaborated logic. But there are disadvantages as well. First, FOL is a mathematical logic. Its development was motivated by the program of logistic in mathematics. FOL is thus well suited for describing algebraic structures. But agents do not communicate in terms of algebraic structures. Moreover, Formulas of FOL express only assertions. However, queries and requests are also valid messages. Thus SL defines so-called identifying expressions. In SL we can also express propositional attitudes, i.e. assertions about other assertions like "John believes that it is raining.". However, these attitudes are dealt with as relations of a believer to a piece of syntax.

[1] For details on FIPA SL, see `http://www.fipa.org/specs/fipa00008`; for KIF, Knowledge Interchange Format, see `http://www-ksl.stanford.edu/knowledge-sharing/kif/`.

SL is well defined syntactically, but problems appear when one wants to know its semantics. There is no proper specification of semantics in the standardization document; only the section "Notes on FIPA SL Semantics" can be found, which is (as it says) just notes. The standard counts upon well-known semantics of FOL, but due to numerous extensions it is not always applicable. The lack of semantics can have very unpleasant consequences. Two agents relying completely on the standard can understand the same message differently and that can lead to serious misunderstandings between the agents. This is in a direct conflict with the name – "Semantic Language".

Another problem connected with the FIPA SL is that SL content is in general not interconnected with the semantic web. On the other hand, in our TIL-Script language we developed an interconnection between OWL concepts and TIL-Script.

For these reasons we propose the TIL-Script language to be used as a FIPA ACL content language.

4 Transparent Intensional Logic (TIL)

Transparent Intensional Logic (TIL) is the logical system founded by Pavel Tichý (see [7]). It was designed to provide a fine-grained semantics for natural language. Thus TIL is the system of a great expressive power applicable also and primarily to non-mathematical, i.e. empirical domains. As an expressive semantic tool it has a great potential in artificial intelligence and any other area where both computers and humans are to be dealt together. More about the role of logic in artificial intelligence can be found in [6].

TIL has the capacity of capturing almost all the semantic features of natural language. It includes temporal and modal attitudes, epistemic logic, knowledge representation (even modeling knowledge of resource bounded agents) and dynamic aspects of these.

Here we are going to introduce TIL just briefly. For details see [1] and [2].

4.1 Basic Notions

The fundamental notion of TIL is the notion of *construction*. Constructions are analogical to formulas and terms of traditional logics, but there are several fundamental differences. Most logics make a strict border between semantics and syntax. Formulas are used to be defined as well formed sentences of some formal language. That means they are mere strings of characters, and an interpretation is needed to reveal their meaning (semantics).

Constructions are not language expressions (strings of characters). They are abstract procedures, i.e., algorithmically structured objects. Since constructions are themselves semantic objects they do not need to be interpreted and they contain both semantic and syntactic components.

4.2 Types

Constructions are coded (and presented) in a language, which is formally derived from the language of typed lambda-calculus. However, terms of a lambda calculus are not constructions themselves; they are just forms of presentation of constructions. All the entities (including constructions) receive a type in TIL. Again, types are not strings of characters; rather they are objective collections of particular objects. For a type α we denote its elements 'α-objects'.

The infinite hierarchy of types in TIL arises from a *type base*. Type base is a (finite) set of basic *(atomic) types*. For the purpose of a natural language analysis, the standard type base of TIL is an epistemic base containing the following types:

o – *Truth values*. The type consisting of two elements: True (**T**) and False (**F**).
ι – *Individuals*. Simple, bare individuals: the 'lowest-level bearers of properties'.
τ – *Time points*. This type is just the set of real numbers.
ω – *Possible worlds*. The collection of all logically possible states of the world.

Over the basic atomic types *molecular types* are defined as the functional closure over the atomic types. The collection $(\alpha\, \beta_1 \ldots \beta_n$ of all (partial) functions mapping types $\beta_1, \ldots, \beta_n$ to a type α is a type. These types not involving constructions are called *types of order 1*. Since TIL constructions are objects *sui generis* and thus receive a type, we need to define higher-order types as well. However, first we define constructions.

4.3 Constructions

Constructions are the fundamental building blocks of TIL. Depending on valuation v, any construction $v-$constructs an object of some type, or it is v-improper (does not v-construct anything). TIL is a logic of partial functions. There are two kinds of constructions, *atomic* and *molecular*. Atomic constructions do not contain any other constituents but themselves. They supply objects on which compound constructions operate. There are two atomic constructions:

Trivialization ${}^0X$ is an elementary construction constructing the object Xa in the simplest way, without a mediation of any other construction.

Variable x is a construction ('x' is just a name). It constructs an object of a respective type dependently on a valuation v; it v-constructs.

Molecular (multiple-step) constructions are:[2]

Composition $[FC_1 \ldots C_n]$ is an instruction to apply a function to its arguments. If F v-constructs a function f of type $(\alpha\beta_1 \ldots \beta_n)$, and C_i v-construct objects c_i of type β_i, then the Composition v-constructs the value of f at the tuple-argument $\langle c_1,\ldots,c_n\rangle$. Otherwise the Composition is $v-$improper.

Closure $[\lambda x_1 \ldots x_n \quad C]$ is an instruction to construct a function in the manner of lambda calculi. If variables $x_1,\ldots,x_n$ range over $\beta_1, \ldots, \beta_n$, respectively, and C v-constructs an object of type α, Closure v-constructs the following function

[2] And two others, Execution and Double Execution, which we are not going to use here.

f: let v' be a valuation that associates x_i with b_i and is identical to v otherwise. Then f is undefined on $b_1, \ldots, b_n$ if C is v' improper, otherwise the value of f on $b_1, \ldots, b_n$ is what is v'-constructed by C.

4.4 Higher order types

Each construction is of some order. The order of a construction is the highest order of the type of objects constructed by sub-constructions of the given construction. Thus the basic type of order 2 is the type $*_1$ – the collection of all constructions of order 1 which v-construct non-constructional entities belonging to a type of order 1. The type $*_2$ is the type of constructions of order 2 $v-$constructing entities belonging to a type of order 2 or 1. And so on, *ad infinitum.* Any type of order n is also a type of order $n + 1$ (type rising). Other types of order n are functional closures $(\alpha\beta_1 \ldots \beta_n)$ of defined types as specified in Section 4.2.

5 TIL-Script as a Content Language

Transparent Intensional Logic is a suitable logic system for utilization as a content language in multiagent systems. For this purpose its main advantages arise from the following TIL features:

Semantic nature Constructions of TIL are themselves semantics objects. So the semantics is naturally well defined. There is no danger of misunderstandings as with the SL language.

High expressibility The expressive power of TIL is really high. TIL is capable of analyzing almost any semantic feature of natural languages.

Original purpose TIL unlike mathematical logics was intended to be a tool for logical analysis of language. Primarily it was designed for natural languages, but this gives it a great potential even in other areas.

The TIL-Script language has been described in [8] and [9].

5.1 Ontologies for TIL-Script

OWL based Ontologies Any content language is tightly related to ontologies. All concepts used or mentioned by a content language must be defined in an ontology. And vice versa, the content language must be able to use any concept from the ontology. FIPA definition of ontology is relatively vague. It just says that ontology provides a vocabulary of domain specific concepts and relations between them. This leads to diversity in implementations. Actually ontology takes a frame-like structure, which is well suitable for the FIPA SL language and developer frameworks like Jade support it.

The recent trend is to use well-proven technologies of semantic web, in particular the OWL language, for defining ontologies. But the existing implementation tools for multi-agent systems do not support OWL very well. The way we have chosen for TIL-Script is to inter-connect the language with

frame-like ontologies because of an implementation in Jade. Integration of OWL into TIL-Script is a subject of our recent research.

Concepts (or classes) of ontologies are sets of so-called individuals. We must not confuse these individuals with members of the TIL-Script type Indiv. Ontology individuals can be objects of any TIL-Script type. For TIL-Script this means that any ontology concept (class) which members are of type α is an object of type $(o\alpha)$, i.e. a set of α-objects. Ontology individuals (members of classes) are directly α-objects.

Inter-connection of TIL-Script with an ontology is mediated by the Trivialization construction. You may Trivialize any object or individual defined by the ontology. However, objects defined in the ontology cannot be Trivialised in TIL-Script. The only demand for an ontology to be used together with TIL-Script is that any class must have defined the TIL-Script type of its members.

GIS based Ontologies Geographic information systems (GIS) are generally used for gathering, analysis and visualization of information on the space aspects of real-world objects. The main advantage of GIS is the ability to relate different kind of information obtained from different sources of a spatial context. This feature enables agents to act in the real-world environment and make decisions based on its state. Agents usually dwell in the virtual world of a computer memory and they are not aware of their position, or of the objects and other agents around. GIS ontology enables agents to receive, manipulate and exchange spatial information.

Ontologies were developed to facilitate knowledge representation, sharing and exchanging. Spatial agents make use of geographic information as the input source for the decision-making mechanism. Situated agents are context-aware. They are aware of their position in space, actual speed, surrounding objects and relationships between them.

Agent actions are usually modelled by defining their behaviours. Each agent can perform the limited amount of predefined actions which can be combined into more complex behaviours. Creating ontology is then divided into two steps.

1. Identification and definition of agent behaviours.
2. Creating behaviour-specific ontology.

Agents can perceive their environment by sensors or geographic database and transform it into the symbolic representation of ontology.

Communication Reconstruction and Knowledge Base Summarizing the features introduced up to now, we can now present an agents' communication scheme based on FIPA standards using Knowledge base and TIL-Script as a content language. Figure 2 illustrates such a schema.

Example of a Knowledge Base:
Concept: "Nissan Skyline GTS-R"
Instance: *The* Nissan Skyline GTS-R, which I am parking outside.

In the TIL-Script language we have two separate ontologies. First, speech-act ontology is connected with the operative part of an agent. Agents are "born" with a minimal ontology in order to be able to communicate with its surroundings and learn by experience.

Second ontology that contains application domain is also replicated to an agent knowledge base. It instantiates its knowledge base and all concepts and roles are copied into it. It means that every agent has equal basic abilities at the time of their creation.

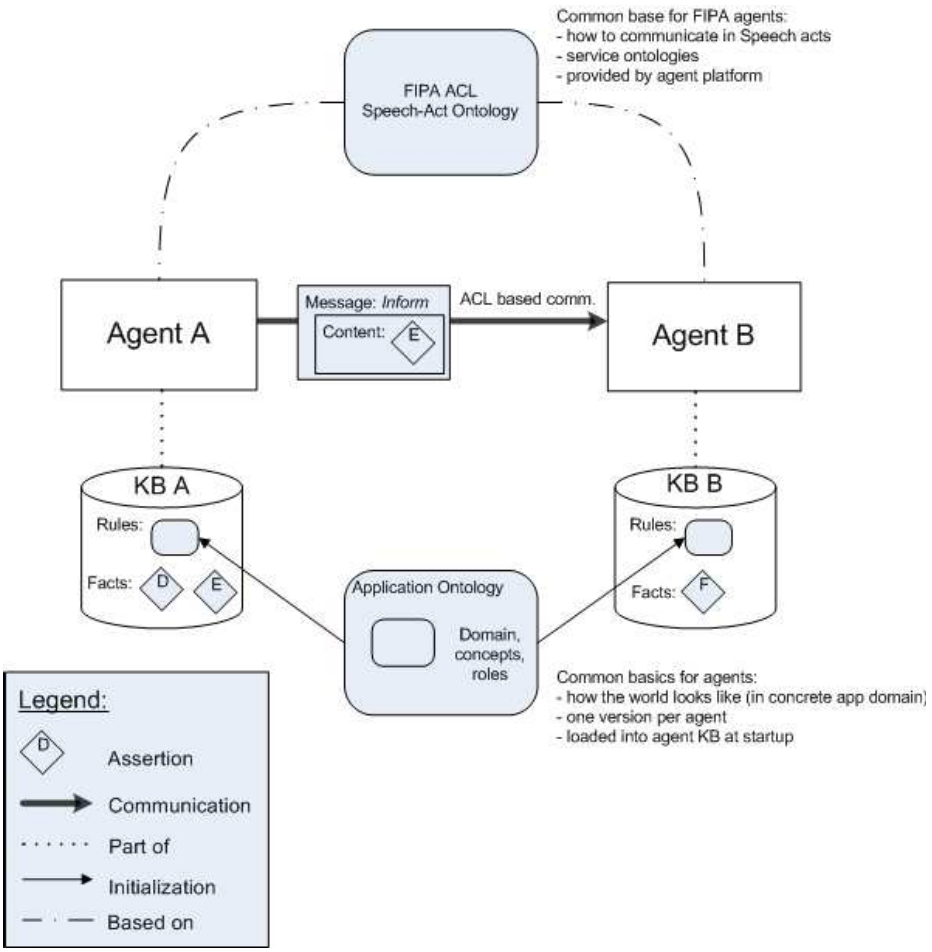


Fig. 2. FIPA Inform message with TIL-Script in the content field

6 Example

In this section we present a simple example scenario of communication of two agents in a multi-agent system using the TIL-Script language.

Scenario. The situation is simple. There is a large car parking-site, a railway station and two agents:

- Driver; an agent driving a car, who wants to park at the parking lot rather close to the railway station.
- Dis*patcher*; a dispatcher of the parking-site managing the pull-ins.

A sketch of their communication is as follows:

Driver: I want you to park me somewhere near to the railway station.

Dispatcher: OK, I can park you at this pull-in (concrete id).

Driver: I accept.

Ontology. In order to analyze the communication using TIL-Script we need to specify ontology of the used concepts.

`TheDriver/Indiv` – The driver agent.

`TheDispatcher/Indiv` – The dispatcher agent.

`TheTrainStation/Indiv` – The train station the driver wants to park close to.

`Pull-in/((Bool Indiv)Time)World` – A property to be a pull-in at a car-park. The pull-in has a slot Id, which is a number indentifying a concrete pull-in.

`Near/((Bool Indiv Indiv)Time)World` – A binary predicate saying that two places are near (at a given time and state of the world).

`Arrange/(Bool (Indiv ((Bool)Time)World))` – This predicate means that an agent need to perform an action. In our conception, an agent (Indiv) has to arrange that a proposition ((Bool)Time)World)) become true.

`Park/(Bool Indiv Indiv)Time)World` – A binary predicate saying that a given object parks at a car-park at a given pull-in.

Communication. Now we can reconstruct the communication between the driver and the dispatcher precisely.

Driver. Call for a proposal:

```
\x:Indiv['Arrange 'TheDispatcher \w\t['And ['Park@w,t 'TheDriver
x] ['Near@w,t 'TheTrainStation x]]]
```

Dispatcher. Proposal:

```
[\x:Indiv['Arrange 'TheDispatcher \w\t['And
['Park@w,t 'TheDriver x] ['Near@w,t 'TheTrainStation x]]] [Id_36]]
```

Driver. I accept the proposal: (the content of the message is the same as in proposal, only the performative is "accept proposal").

7 Conclusion

Actual standards for communication in multi-agent systems are based on syntax rather than semantics. This can slow down the progress in future

research. As an option we propose the TIL-Script language, which is based on a well elaborated Transparent Intensional Logic. This makes TIL-Script a semantically based language suitable for communication in multi-agent systems.

High expressive power of TIL-Script makes it also a suitable tool for adopting other logics and languages into its semantic framework, so that TIL-Script can be used as a specification language. A big potential of TIL-Script can be also found in logical analysis of natural languages and communication with human (non-expert) agents.

The TIL-Script language is being implemented and tested in multi-agent systems using the Python language and Jade based simulation programs. Ontology support is designed for frame-like ontologies supported by Jade. Using OWL ontologies supported by Protége has been developed and tested as a separate part of the research project. Storing ontologies into Knowledge Base is arranged with SQLite DataBase Management System (it behaves as a client application so that there is no centralized component in the system) with application layer implemented in Python (Jython).

Acknowledgements. This research has been supported by the program 'Information Society' of the Czech Academy of Sciences within the project "Logic and Artificial Intelligence for multi-agent systems", No. 1ET101940420.

References

1. Duží, M., Jespersen, B. and Muller, J. Epistemic closure and inferable knowledge. In Libor Běhounek and Marta Bílková, (Eds.), the *Logica Yearbook 2004*. Prague: Filosofia, 2005.
2. Duží, M. and Materna, P. *Constructions*. Retrievable at `http://til.phil.muni.cz/text/constructions_duzi_materna.pdf`, 2000.
3. Foundation for Intelligent Physical Agents. FIPA Abstract Architecture Specification. `http://fipa.org/specs/fipa00001`, 2000.
4. Foundation for Intelligent Physical Agents. FIPA SL Content Language Specification. `http://fipa.org/specs/fipa00008`, 2000.
5. Luck, M., McBurney, P., Shehory, O. and Willmott, S. Agent Technology: Computing as Interaction. A Roadmap for Agent-Based Computing. *AgentLink*, 2005.
6. Thomason, R. *Logic and artificial intelligence*. In: Edward N. Zalta, (Ed.): The Stanford Encyclopedia of Philosofy. Summer 2005.
7. Tichý, P. *The Foundations of Frege's Logic*. Walter de Gruyter, Berlin-New York, 1988.
8. Ciprich, N., Duží, M., Košinár, M.: *TIL-Script: Functional Programming Based on Transparent Intensional Logic*. In: *RASLAN 2007*, Sojka, P., Horák, A., (Eds.), Masaryk University Brno, 2007, pp. 37–42.
9. Ciprich, N., Duží, M. and Košinár, M.: *The TIL-Script language*. To appear in the Proceedings of the 18th European Japanese Conference on Information Modelling and Knowledge Bases (EJC 2008), Tsukuba, Japan 2008.

Can Complex Valency Frames be Universal?

Karel Pala and Aleš Horák

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

Abstract. This paper deals with the comprehensive lexicon of Czech verbs named VerbaLex. It contains complex valency verb frames (CVFs) including both surface and deep valencies.
The most notable features of CVFs include two-level semantic labels with linkage to the Princeton and EuroWordNet Top Ontology hierarchy and the corresponding surface verb frame patterns capturing the morphological cases that are typical of the highly inflected languages like Czech.
We take the position that CVFs can be suitable for a description of the argument predicate structure not only of Czech verbs but also verbs in other languages, particularly Bulgarian, Romanian and English.

1 Introduction

Semantic role annotation is usually based on the appropriate inventories of labels for semantic roles (deep cases, arguments of verbs, functors, actants) describing argument predicate structure of verbs. It can be observed that the different inventories are exploited in different projects, e.g. Vallex [1], VerbNet [2], FrameNet [3], Salsa [4], CPA [5], VerbaLex [6].

With regard to the various inventories a question has to be asked: how adequately they describe semantics of the empirical lexical data as we can find them in corpora? From this point of view it can be seen that some of the inventories are rather syntactic than semantic (e.g. Vallex 1.0 or VerbNet). If we are to build verb frames with the goal to describe real semantics of the verbs then we should go 'deeper'. Take, e.g. verbs like *drink* or *eat*, – it is obvious that the role PATIENT that is typically used with them labels cognitively different entities – BEVERAGES with *drink* and FOOD with *eat*. If we consider verbs like *see* or *hear* we can observe similar differences not mentioning the fact that one can see anything. This situation can be improved if we use more detailed subcategorization features which, however, in other lexicons (e.g. in VerbNet or Vallex 1.0) are exploited only partially. If we are not able to discriminate the indicated semantic distinctions the use of the frames with such labels in realistic applications can hardly lead to the convincing and reliable results.

These considerations led us to design the inventory of two-level labels which are presently exploited for annotating semantic roles in Czech verb valency frames in lexical database VerbaLex containing now approx. 10,500 Czech verb lemmata and 19,500 frames.

Petr Sojka, Aleš Horák (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2008, pp. 41–48, 2008.

1.1 Thematic Roles and Semantic types

A question may be asked what is the distinction between “shallow” roles such as AGENT or PATIENT and “deep” roles such as SUBS(food:1), as we use it in VerbaLex. We have already hinted that "shallow" roles seem to be very similar to syntagmatic functions. At the same time it should be obvious that information that a person functions as an agent who performs an action is not only syntagmatic. That was the main reason why we included them in our list of the roles. We do not think that SUBS(food:1) is a special case of the deep role, rather, we would like to speak about a two-level role consisting of the ontological part, i.e. SUBS(tance), and the subcategorization feature part, e.g. beverage:1 which is also a literal in PWN 2.0 that can be reached by traversing the respective hyperonymy/hyponymy tree.

In the Hanks’ and Pustejovsky’s Pattern Dictionary[1] a distinction is made between semantic roles and semantic types: “the semantic type is an intrinsic attribute of a noun, while a semantic role has the attribute thrust upon it by the context.” Also lexical sets are distinguished as “clusters of words that activate the same sense of a verb and have something in common semantically.”

Introduction of the mentioned notions is certainly very inspiring in our context, however, we think that at the moment the quoted ’definitions’ as they stand do not seem to be very operational, they are certainly not formal enough for computational purposes. What is needed are the lists of the semantic roles and types but they are being created gradually along with building the necessary ontology. Thus for time being we have to stick to our two-level roles as they are. They are partly based on the TOP Ontology as used in EuroWordNet project [8].

2 VerbaLex and Complex Valency Frames

The design of VerbaLex verb valency lexicon was driven mainly by the requirement to describe the verb frame (VF) features in a computer readable form that could be used in the course of automatic syntactic and semantic analysis. After reviewing actual verb frame repositories for Czech, we have decided to develop *Complex Valency Frames* (CVFs) that contain:

- morphological and syntactic features of constituents,
- two-level semantic roles,
- links to PWN and Czech WordNet hypero/hyponymic (H/H) hierarchy,
- differentiation of animate/inanimate constituents,
- default verb position,
- verb frames linked to verb senses,
- VerbNet classes of verbs.

[1] cf. [5] and also [7]

3 Role Annotation and EWN Top Ontology

Presently, our inventory contains about 30 general or ontological labels selected from the EuroWordNet Top Ontology (EWN TO), with some modifications, and the 2nd-level subcategorization labels taken mainly from the Set of Base Concepts introduced in EuroWordNet Project (1999). The 2nd-level labels (approx. 200) selected from the Set of Base Concepts (BCs) are more concrete and they can be viewed as subcategorization features specifying the ontological labels obtained from EWN TO. The motivation for this choice is based on the fact that WordNet has a hierarchical structure which covers approx. 110,000 English lexical units (synsets). It is then possible to use general labels corresponding to selected top and middle nodes and go down the hyperonymy/hyponymy (H/H) tree until the particular synset is found or matched. This allows us to see what is the semantic structure of the analyzed sentences using their respective valency frames. The nodes that we have to traverse when going down the H/H tree at the same time form a sequence of the semantic features which characterize meaning of the lexical unit fitting into a particular valency frame. These sequences can be interpreted as detailed selectional restrictions.

Two-level labels contain ontological labels taken from EWN TO (about 30) that include roles like AGENT, PATIENT, INSTRUMENT, ADDRESSEE, SUBSTANCE, COMMUNICATION, ARTIFACT at the 1st level. The 2nd-level labels that are combined with them are literals from PWN 2.0 together with their sense number.

The nice property of the Czech valency frames is that the semantic restrictions are endogenous, i.e. they are specified in terms of other synsets of the same WordNet.

The notation allows us to handle basic metaphors as well. An example of CVFs for *drink/pít* may take the form:

```
who_nom*AGENT(human:1|animal:1) <drink:1/pít:1>
   what_acc*SUBS(beverage:1)
```

4 Can CVFs be Universal?

The building VerbaLex database started during the EU project Balkanet (Balkanet Project, 2002) when about 1,500 Czech verb valency frames were included in Czech verb synsets. They were linked to English Princeton WordNet and to the WordNets of other languages in Balkanet by means of the Interlingual Index (ILI). We tested a hypothesis that the Czech complex valency frames can be reasonably applied also to the verbs in other languages, particularly to Bulgarian and Romanian. Thus, in the Balkanet project an experiment took place in which CVFs developed for Czech verbs have been adopted for the corresponding Bulgarian and Romanian verb synsets [9,10]. The results of the experiments were positive (see below the Section 4.1), therefore a conclusion can be made that this can be extended also for other languages.

The question then remains whether CVFs developed for Czech can be applied to English equally well. If we exploit ILI and have look at the VFs for Czech/English verbs like *pít/drink, jíst/eat* and apply them to their English translation equivalents we come to the conclusion that the Czech deep valencies describe well their semantics. VerbaLex is incorporated into Czech WordNet and through ILI also to PWN 2.0, thus we have the necessary translation pairs at hand. This also can be applied to other WordNets linked to PWN v.2.0. Thus we rely on the principle of translatability which means that the deep valencies developed for Czech verbs can be reasonably exploited also for English (see the Section 3). There is a problem with surface valencies which in English are based on the fixed order SVOMPT and on morphological cases in Czech but we consider this rather a technical issue at the moment.

4.1 Bulgarian example

The enrichment of Bulgarian WordNet with verb valency frames was initiated by the experiments with Czech WordNet (CzWN) which, as we said above, already contained approx. 1,500 valency frames (cf. [11]). Since both languages (Czech and Bulgarian) are Slavonic we assumed that a relatively large part of the verbs should realize their valency in the same way. The examples of Bulgarian and Czech valency frames in the Figure 1 show that this assumption has been justified (English equivalents come from PWN 1.7).

The construction of the valency frames of the Bulgarian verbs was performed in two stages:

1. Construction of the frames for those Bulgarian verb synsets that have corresponding (via Interlingual Index number) verb synsets in the CzWN and in addition these CzWN synsets are provided with already developed frames.
2. Creation of frames for verb synsets without analogues in the CzWN. The frames for more than 500 Bulgarian verb synsets have been created and the overall number of added frames was higher than 700. About 25 % of the Bulgarian verb valency frames we used without any changes, they match the Czech ones completely.

In our view the Bulgarian experiment is convincing enough and it shows sufficiently that it is not necessary to create the valency frames for the individual languages separately.

4.2 Romanian example

D. Tufis et al [10] investigated the feasibility of the importing the valency frames defined in the Czech WordNet [12] into the Romanian WordNet. They simply attached Czech valency frames from CzWn to the Romanian verbs. As we hinted above the Czech CVFs specify syntactic and semantic restrictions on the predicate argument structure of the predicate denoting the meaning of a

produce, make, create – create or manufacture a man-made product
BG: {proizveždam} njakoj*AG(person:1)| neščo*ACT(plant:1)= neščo*OBJ(artifact:1)
CZ: {vyrábět, vyrobit} kdo*AG(person:1)| co*ACT(plant:1) = co*OBJ(artifact:1)
uproot, eradicate, extirpate, exterminate – destroy completely, as if down to the roots; "the vestiges of political democracy were soon uprooted"
BG: {izkorenjavam, premachvam} njakoj*AG(person:1)| neščo*AG(institution:2)= neščo*ATTR(evil:3)|*EVEN(terrorism:1)
CZ: {vykořenit, vyhladit, zlikvidovat} kdo*AG(person:1)|co*AG(institution:2) = co*ATTR(evil:3)|*EVEN(terrorism:1)
carry, pack, take – have with oneself; have on one's person
BG: {nosja, vzimam} njakoj*AG(person:1)= neščo*OBJ(object:1)
CZ: {vzít si s sebou, brát si s sebou, mít s sebou, mít u sebe} kdo*AG(person:1)= co*OBJ(object:1)

Fig. 1. Common verb frame examples for Czech and Bulgarian

given synset. The valency frames also specify the morphological cases of the arguments. Let us consider, for instance, the Romanian verbal synset ENG20-02609765-v (a_se_afla:3.1, a_se_g'asi:9.1, a_fi:3.1) with the gloss "be located or situated somewhere; occupy a certain position." Its valency frame is described by the following expression:(nom*AG(fiint'a:1.1)| nom*PAT(obiect_fizic:1)) = prep-acc*LOC(loc:1).

The specified meaning of this synset is: an action the logical subject of which is either a fiint'a (sense 1.1) with the AGENT role(AG), or a obiect_fizic (sense 1) with the PATIENT role (PAT). The logical subject is realized as a noun/NP in the nominative case (nom). The second argument is a loc (sense 1) and it is realized by a prepositional phrase with the noun/NP in the accusative case (prep-acc). Via the interlingual equivalence relations among the Czech verbal synsets and Romanian synsets we imported about 600 valency frames. They were manually checked against the BalkaNet test-bed parallel corpus (1984) and more than 500 complex valency frames were found valid as they were imported or with minor modifications. This result supported by the real evidence is more than promising. Czech CVFs also motivated Tufis' group for further investigations on automatically acquiring FrameNet structures for Romanian and associating them with WordNet synsets.

4.3 English example

Let us take the complex valency frame for the Czech verb *učit se (learn)* and its deep valency part describing the basic meaning of the verb:

`kdo1*AG(person:1)=co4*KNOW(information:3)[kde]*GROUP(institution:1)`
(ex.: *učit se matematiku ve škole – to learn mathematics in the school*)

If the translation pair *učit se – learn* is correct then we can conclude that this frame is suitable both for Czech and English.

Similarly, take the Czech and English verb *pít/drink* with their basic meaning again. The relevant deep part of the CVFs takes the following shape:

`kdo1*AG((person:1)|(animal:1))=co4*SUBS(beverage:1)`
(ex.: *bratr pije pivo, kůň pije vodu – my brother drinks beer, the horse drinks water*)

Again, it can be seen that this CVF describes well both Czech verb meaning and the meaning of its English translation equivalent.

It may be argued that these are only two examples and there may be some doubtful cases. When linking Czech and English verb synsets via ILI is finished more examples can be adduced to show that the CVFs can serve for Czech and English equally well not speaking about other languages. To get more necessary evidence we are going to examine first selected semantic classes of the Czech and English verbs (see the next section), such as verbs of drinking, eating, verbs denoting animal sounds, putting, weather. Even brief comparison shows that their CVFs appear suitable for both languages and not only for them.

In VerbaLex we presently have about 10,500 Czech verb lemmata. From them approx. only 5,158 have been linked to the Princeton WordNet 2.0 via ILI earlier. After processing all VerbaLex verbs we have linked to Princeton WordNet further 3,686 Czech verbs. Altogether 8,844 Czech verbs are now linked to Princeton WordNet. The processing of the VerbaLex verbs and their linking to PWN v.2.0 shown however, that approx. 15 % of the Czech verb synsets cannot be linked to PWN v.2.0 since it is not possible to find their lexicalized translation equivalents in English. It should be remarked that this is a serious problem which, however, has to be solved separately.

5 Semantic Classes of Czech Verbs

We have worked out semantic classes of Czech verbs that were inspired by Levin's classes [13] and VerbNet classes [2]. Since Czech is a highly inflectional language the patterns of alternations typical for English cannot be straightforwardly applied – Czech verbs require noun phrases in morphological cases (there are 7 of them both in singular and plural). However, classes similar to Levin's can be constructed for Czech verbs as well but they have to be based only on the grouping of the verb meanings. Before starting the VerbaLex project we had compiled a Czech-English dictionary with Levin's 49 semantic classes and their Czech equivalents containing approx. 3,000 Czech verbs.

In VerbaLex project we went further and tried to link Czech verbs with the verb classes as they are used in VerbNet – they are also based on Levin's classification extended to almost 400 classes. This meant that for each Czech verb in VerbaLex we had marked the VerbNet semantic class a verb belongs to. The next step, however, was to have a look at the semantic roles introduced in VerbaLex. This led us to the reduction of the VerbNet semantic classes to 89 – the semantic roles helped us to make the semantic classification of the verbs more consistent. For example, if we take semantic role BEVERAGE – it is yielding a homogeneous group containing 62 verbs. It can be seen that Levin's classes sometimes contain verbs that seem to form one consistent group but if we look at them closer it becomes obvious that they inevitably call for further

subclassification. For instance, if we take the class containing verbs of putting (put-9 in VerbaLex notation) we can see that it contains verbs like *to put* on one hand, but also *to cover* or *to hang* on the other. These differences in their meaning have to be captured.

The basic assumption in this respect is that there is a mutual relation between semantic classes of verbs and the semantic roles in their corresponding CVFs. In this way both the consistency of the inventory of semantic roles and consistency of the related semantic verb classes can be checked – obviously, in one class we can expect the roles specific only for that class. For example, with verbs of clothing the role like GARMENT with its respective subcategorizations reliably predicts the respective classes and their subclasses. Similarly it works for other verb classes, such as verbs of eating (FOOD), drinking (BEVERAGE), emotional states, weather and many others.

In our view, the good news also is that if the semantic parts of the CVFs can work for more languages (as we tried to show) the same can be extended for the corresponding semantic verb classes.

The ultimate goal is to obtain semantic verb classes suitable for further computer processing and applications.

6 Conclusions

In the paper we deal with the lexical database of Czech verbs VerbaLex whose main contribution consists in the development complex valency frames (CVFs) capturing the surface (morphological) and deep (semantic) valencies of the corresponding verbs. For labeling the roles in the valency frames we have worked out a list (ontology) of the two-level semantic labels which at the moment contains approx. 30 'ontological' roles and 200 subcategorization features represented by the literals taken from Princeton WordNet 2.0. At present VerbaLex contains approx. 10,500 Czech verbs with 19,000 CVFs. From them

Further, we pay attention to some multilingual implications and show that originally 'Czech' Complex Valency Frames can reasonably describe semantics of the predicate argument structures of Bulgarian, Romanian and English verbs and obviously also verbs in other languages. What has to be dealt with separately are surface valencies because they heavily depend on the morphological cases in Czech and Bulgarian and syntactic rules of Romanian and English. The issue calls for further testing and validation, however, we consider the presented analysis more than promising.

Acknowledgments. This work has been partly supported by the Academy of Sciences of Czech Republic under the project 1ET100300419 and by the Ministry of Education of CR in the National Research Programme II project 2C06009 and project LC536.

References

1. Straňáková-Lopatková, M., Žabokrtský, Z.: Valency Dictionary of Czech Verbs: Complex Tectogrammatical Annotation. In: LREC 2002, Proceedings. Volume III, ELRA (2002) 949–956.
2. Kipper, K., Dang, H.T., Palmer, M.: Class Based Construction of a Verb Lexicon. In: AAAI-2000 17th National Conference on Artificial Intelligence, Austin TX (2000).
3. Fillmore, C., Baker, C., Sato, H.: Framenet as a 'net'. In: Proceedings of Language Resources and Evaluation Conference (LREC 2004). Volume 4., Lisbon, ELRA (2004) 1091–1094.
4. Boas, H.C., Ponvert, E., Guajardo, M., Rao, S.: The current status of German FrameNet. In: SALSA workshop at the University of the Saarland, Saarbrucken, Germany (2006).
5. Hanks, P.: Corpus Pattern Analysis. In: Proceedings of the Eleventh EURALEX International Congress, Lorient, France, Universite de Bretagne-Sud (2004).
6. Hlaváčková, D., Horák, A.: VerbaLex – New Comprehensive Lexicon of Verb Valencies for Czech. In: Proceedings of the Slovko Conference, Bratislava, Slovakia (2005).
7. Hanks, P., Pala, K., Rychlý, P.: Towards an empirically well-founded semantic ontology for NLP. In: Workshop on Generative Lexicon, Paris, France (2007) in print.
8. Vossen, P., ed.: EuroWordNet: A Multilingual Database with Lexical Semantic Networks. Kluwer Academic Publishers, Dordrecht (1998).
9. Koeva, S.: Bulgarian VerbNet. Technical Report part of Deliverable D 8.1, EU project Balkanet (2004).
10. Tufis, D., Barbu, Mititelu, V., Bozianu, L., Mihaila, C.: Romanian WordNet: New Developments and Applications. In: Proceedings of the Third International WordNet Conference – GWC 2006, Jeju, South Korea, Masaryk University, Brno (2006) 336–344.
11. Koeva, S., et al.: Restructuring wordnets for the balkan languages, design and development of a multilingual balkan wordnet balkanet. Technical Report Deliverable 8.1, IST-2000-29388 (June 2004).
12. Pala, K., Smrž, P.: Building the Czech Wordnet. In: Romanian Journal of Information Science and Technology. 7(2–3), (2004) 79–88.
13. Levin, B., ed.: "English Verb Classes and Alternations: A Preliminary Investigation". The University of Chicago Press, Chicago (1993).

Processing Czech Verbal Synsets with Relations to English WordNet

Vašek Němčík, Dana Hlaváčková, Aleš Horák, Karel Pala, and Michal Úradník

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

Abstract. In the paper, we present the latest results of the process of final work on preparing the second stable version of the large lexicon of Czech verb valencies named VerbaLex.
VerbaLex lexicon is developed at the Masaryk University NLP Centre for the last three years. The current stage of this unique language resource aims at full interconnection with the pivot world semantic network, the Princeton WordNet. The paper describes the techniques used to achieve this task in a semi-automatic way.
We also describe interesting parts of preparation of a printed representative lexicon containing an important subset of VerbaLex with the valency description adapted to human readable forms.

1 Introduction

One of the most difficult tasks of automatic language processing is the analysis of meaning of language expressions or sentences. The central point of every sentence analysis is formed by its predicate part, i.e. by the analysis of the *verb* and *its arguments*. Within the process of design and development of the Czech syntactic analyser [1], this assumption has led us to the creation of a representative lexicon of Czech verbs and verb valency frames containing many pieces of information suitable for computer processing of the verb arguments from the point of view of their syntax as well as semantics (related to the content of the central semantic network of English, the Princeton WordNet [2]).

In the current text, we show the details of discovering relations between more than three thousands of new Czech verbal synsets and the respective English synsets, which finally allows to incorporate all the VerbaLex synsets to one of its original sources, the Czech WordNet [3].

2 VerbaLex Valency Lexicon

VerbaLex is a large lexical database of Czech verb valency frames and has been under development at The Centre of Natural Language Processing at the Faculty of Informatics Masaryk University (FI MU) since 2005. The organization

Petr Sojka, Aleš Horák (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2008, pp. 49–55, 2008.

of lexical data in VerbaLex is derived from the WordNet structure. It has a form of synsets arranged in the hierarchy of word meanings (hyper-hyponymic relations). For this reason, the headwords in VerbaLex are formed by lemmata in synonymic relations followed by their sense numbers (standard Princeton WordNet notation).

The basic valency frames (BVF) with stressed verb position contain various morpho-syntactic and semantic information. The types of verbal complementation (nouns, adjectives, adverbs, infinitive constructions or subordinate clauses) are precisely distinguished in the verb frame notation. The type of valency relation for each constituent element is marked as obligatory (obl) or optional (opt). BVF is followed by an example of verb usage in a sentence. Semantic information of verbal complementation is represented by two-level semantic roles in BVF. The first level contains the main semantic roles proposed on the 1st-order Entity and 2nd-order Entity basis from the EuroWordNet Top Ontology. The 1st-level semantic roles represent a closed list of 30 semantic tags. On the second level, we use specific selected literals (lexical units) from the set of Princeton WordNet Base Concepts with the relevant sense numbers. We can thus specify groups of words (hyponyms of these literals) suitable for relevant valency frames. This concept allows us to specify valency frames notation with a large degree of sense differentiability. The list of 2nd-level semantic roles is open, the current version contains about 1,000 WordNet lexical unites.

VerbaLex captures additional information about the verbs, which is organized in *complex valency frames* (CVF):

- definition of verb meanings for each synset;
- verb ability to create passive form;
- number of meaning for homonymous verbs;
- semantic classes;
- aspect;
- types of verb use;
- types of reflexivity for reflexive verbs.

The current version of VerbaLex contains 6,360 synsets, 21,193 verb senses, 10,482 verb lemmata and 19,556 valency frames. The valency database is developed in TXT format and available in XML, PDF and HTML formats.

3 Linking New VerbaLex Synsets to WordNet

Extending a valency lexicon such as VerbaLex is a complex task. It does not only consist in editing valency frames of the individual verbs, but comprises also linking the data to other linguistic resources, and eliminating duplicates and inconsistencies. In the case of VerbaLex, each of the 3,686 newly added synsets is linked to its counterpart in the English Princeton WordNet (PWN) and to its hypernym in Czech WordNet. The linking procedure cannot be automated reliably and requires very costly human lexicographers to do a great amount of humdrum work.

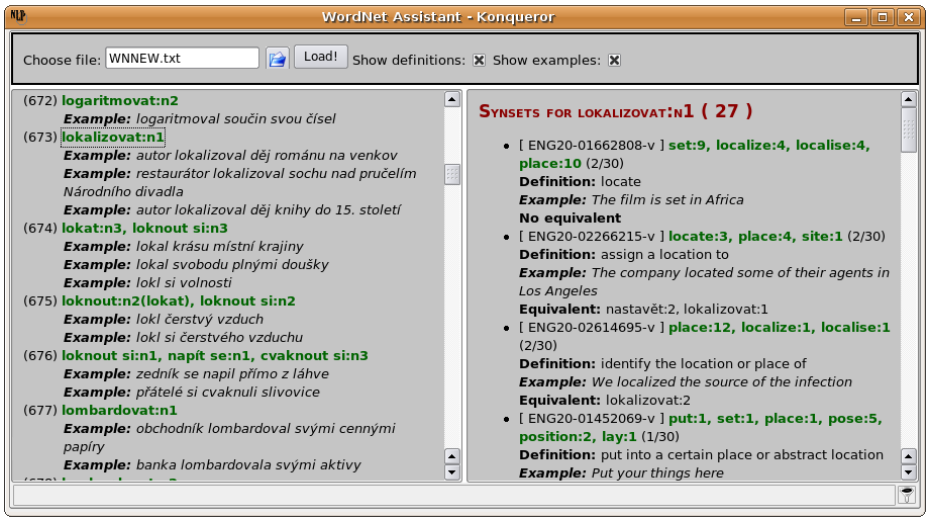


Fig. 1. WordNet Assistant

Based on our earlier experience, when linking new Czech synsets to PWN, the human lexicographers typically look up English translations for the individual synset literals. Subsequently, they use the translations to query the DEBVisDic WordNet browser and search for the corresponding English synset within the query results. To alleviate the human lexicographers from such routine tasks and to speed up the linking process, certain procedures have been semi-automated.

3.1 WordNet Assistant

WordNet Assistant (WNA) is a piece software that carries out certain steps, that would have to be otherwise carried out by a human lexicographer, automatically. This helps the lexicographer concentrate on crucial decisions and thus work more efficiently.

First, WordNet Assistant obtains individual literal translations by looking them up in a multi-lingual dictionary, in our case the GNU/FDL English-Czech Dictionary compiled at the Technical University in Plzeň [4]. Then, it uses these translations to query the English WordNet using the DEB server [5] interface, in order to obtain the relevant English synsets. On top of that, it sorts the English synsets yielded in the previous step according to their estimated relevance. More relevant synsets are presented to the user in a more prominent way (i.e. higher on the list). The user interface of WNA can be seen in Figure 1.

The heuristics used is not entirely accurate. It is based on the assumption that the more translations of literals of the original synset the English synset contains, the more relevant it is.

In addition to the above-mentioned functionality, WNA helps the lexicographer locate the hypernym of the new Czech synset in Czech WordNet. The hypernym can be suggested based on the already determined English counterpart in the following way:

- start at the English synset corresponding to the new Czech synset
- check whether the current synset has a Czech counterpart
- if there is no Czech counterpart, move to the hypernym of the current synset and continue with the previous step
- if there is one, it is the resulting related synset

Like in the case of sorting the English synsets according to their relevance, the suggested synset need not necessarily be the correct one. However, it is more than sufficient when it is close enough to lead the lexicographer to the relevant part of the hyper-hyponymic tree. It seems to be unavoidable anyway that the human lexicographer inspects and compares a number of close synsets before making the linking decision.

Generally, providing a hyper-hyponymic subtree of a reasonable size or a number of synsets, rather than a single one, helps prevent and detect inconsistencies. Given that the English synset may have been already linked with some other Czech synset, the lexicographer may consider revising the linking, merging the Czech synsets in question, or adding some distinctive features that would make it possible to link one of the synsets to a more specific English counterpart.

The work on the final version of linking the current VerbaLex synsets to PWN is reaching its end, however, no precise evaluation is available yet. We plan to analyze the information on the position of the synset chosen by the human lexicographer on the list presented by WNA, and based on that, to study the accuracy of the relevance heuristics. Nevertheless, for our purposes, it is more appropriate to evaluate the system in terms of time saved by the human lexicographers. Certain saving in time contributable to WNA is in principle guaranteed, because the lexicographers need to gather the information computed by WNA anyway.

3.2 Problems in Linking to PWN

The set of synsets newly added to VerbaLex contains a number of not particularly common verbs like “bručet” (“to grumble”) or “nachodit se” (“to have a lot of walking around”), for which it is extremely hard, or even impossible to find an English lexicalized counterpart. These synsets have been marked as “unlinkable”, comprise approximately 15 % of all synsets and can be divided into a number of categories:

- Perfective verbs (usually denoting an end of action)
  doletět (“to arrive somewhere by flying”), *dočesat* (“to finish combing”), *dokrmit* (“to finish feeding”)

- Reflexive verbs
  naběhat se (“to have a lot of running around”), *nalítat se* (“to have a lot of rushing around”), *načekat se* (“to do a lot of waiting”), *maskovat se* (“to disguise oneself”)
- Metaphoric expressions
  nalinkovat (“to line something for somebody” meaning “to plan/determine something for somebody”), *žrát* (“to eat somebody” meaning “to nag at somebody”)
- Expressive verbs
  ňafat se (“to argue”),
- Verbs with no direct English equivalent
  přistavit (“to park/stop a vehicle somewhere”)
- Verbs with no equivalent in PWN
  přepólovat (“to change the polarity of something”)

It should be remarked that similar problems have been already discussed during building the first version of the Czech WordNet in the course of the EuroWordNet project [6]. The issues of translatability between typologically different languages have been also touched in the Balkanet project where the notion of virtual nodes was suggested. They call for a special study.

Further, additional checks have been performed to detect duplicate VerbaLex synsets. Considering the size of the lexicon and the intricacies of Czech, duplicities cannot be completely prevented. Thanks to the linking of VerbaLex to PWN, it is possible to group semantically related synsets and further improve the quality of the data. Such synsets are for example: *baculatět:N1, buclatět:N1* and *kulatět se:N1, kulatit se:N1*. These synsets are both linked to the PWN synset “round:8, flesh out:3, fill out:6”,they are synonymous and need to be merged.

4 Compressed Human Readable Form of Verb Valency Frames

The process of presentation and checking of the verb valency expressions is a complicated task that can be partly done in an automatic manner, but in case of the need of a small subset with 100 % certainty of correctness the work must be done by human linguistic experts. That is why, for the purpose of preparing a printed book of the most frequent and/or somewhat specific verbs in VerbaLex, we have developed and implemented translation of VerbaLex frames to a compressed human readable form, see an example in the Figure 2. Such translation allows a human reader to grasp the meaning of the valency frame(s) in a fast and intuitive way.

The semantic roles in the frames need to be substituted with words from the particular language (Czech) and inflected in the correct form. This substitution is done by translating each role to Czech obtaining the lemmatized form of the role. The required inflected forms of the roles and verbs are then obtained using the morphological analyzer `ajka` [7]. This tool can generate all forms of an input

One VerbaLex frame for *připustit/připouštět* (admit):

```
ORG(person:1)who_nom VERB STATE(state:4)what_acc
GROUP(institution:1)who_nom VERB STATE(state:4)what_acc
ORG(person:1)who_nom VERB STATE(state:4)that
GROUP(institution:1)who_nom VERB STATE(state:4)that
```

The compressed human readable form:

```
V: člověk/instituce - připustí, připouští - stav
```

person/institution – admits – a state

Fig. 2. An example of a translation of the VerbaLex frames to a compressed human readable form.

word with the corresponding grammatical tags. With this process, the original VerbaLex frame from the Figure 2 is translated to:

```
V: člověk - připustí, připouští - stav
V: instituce - připustí, připouští - stav
V: člověk - připustí, připouští - stav
V: instituce - připustí, připouští - stav
```

This form is already suitable for human reading, however, we can see that there are duplicate and near-duplicate lines in the output. The "compression" of these forms of verb frames then follows the procedure:

1. remove exact duplicate frames;
2. frames containing phraseologisms are gathered in a separate set;
3. all remaining frames are compared 1:1 as candidates for joining a tuple in one new frame;
4. the best tuple is joined and forms a new frame
5. repeat from step 3 until no new frame can be created;
6. the result is formed by the compressed frames and the original set of frames with phraseologic phrases.

As a result of this procedure, we will obtain the compressed human readable frame for all the VerbaLex verb frames.

5 Conclusion

Within the final work on preparation of the second extended and improved version of the VerbaLex verb valency lexicon, we have designed and implemented the WordNet Assistant software tool. WordNet Assistant is aimed at supporting lexicographers in discovering new equivalents between (Czech) VerbaLex synsets and (English) Princeton WordNet synsets. In this paper, we present the process and problems in exploring such interlingual relations as well as their reuse for merging of very similar synsets.

Finally, for the purpose of presentation and checking of the valency frames, we have described an implemented tool that can offer a compressed human readable form of the VerbaLex verb frames.

We believe that the presented resources and implemented techniques prove that we have achieved new and valuable results in the description of the meaning of Czech verbs.

Acknowledgments. This work has been partly supported by the Academy of Sciences of Czech Republic under the project 1ET100300419 and by the Ministry of Education of CR within the Center of basic research LC536 and in the National Research Programme II project 2C06009.

References

1. Horák, A.: Computer Processing of Czech Syntax and Semantics. Librix.eu, Brno, Czech Republic (2008).
2. Fellbaum, C., ed.: WordNet. An Electronic Lexical Database. MIT Press, Cambridge (1998).
3. Pala, K., Smrž, P.: Building the Czech Wordnet. Romanian Journal of Information Science and Technology **7**(2–3) (2004) 79–88.
4. Svoboda, M.: GNU/FDL English-Czech dictionary (2008) `http://slovnik.zcu.cz/`.
5. Horák, A., Pala, K., Rambousek, A., Povolný, M.: DEBVisDic – First Version of New Client-Server Wordnet Browsing and Editing Tool. In: Proceedings of the Third International WordNet Conference – GWC 2006, Brno, Czech Republic, Masaryk University (2005) 325–328.
6. Vossen, P., Bloksma, L., Peters, W.: Extending the Inter-Lingual-Index with new concepts. (1999) Deliverable 2D010 EuroWordNet, LE2-4003.
7. Sedláček, R.: Morphemic Analyser for Czech. Ph.D. thesis, Masaryk University, Brno, Czech Republic (2005).

Extraction of Syntactic Structures Based on the Czech Parser Synt

Miloš Jakubíček

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

Abstract. In this paper we describe the usage of the syntactic parser `synt` (developed in the NLP Centre at Masaryk University) to gain information about syntactic structures (such as noun or verb phrases) of common sentences in Czech. These structures are from the analysis point of view usually identical to nonterminals in the grammar used by the parser to find possible valid derivations of the given sentence. The parser has been extended in such a way that enables its highly ambiguous output to be used for extracting those syntactic structures *unambiguously* and gives several ways how to identify them. To achieve this, some previously unused results of syntactic analysis have been evolved leading to more precise morphological analysis and hence also deeper distinction among various syntactic (sub)structures. Finally, we present an application for shallow valency extraction.

1 Introduction

Usually, a derivation tree is presented as the main output of syntactic parsing of natural languages, but currently most of the syntactic analysers for Czech lack precision, i.e. there are more (actually, in some cases up to billions) trees given on the output. However there are many situations in which it is not necessary and sometimes even not desirable to have such derivation trees, may it be information extraction and retrieval, transformation of sentences into a form of predicate(arguments) or shallow valency extraction. In such cases we rather need to process whole syntactic structures in the given sentence, especially noun, prepositional and verb phrases, numerals or clauses. Moreover, so as not to end up with the same problems as with the standard parser output, we need to identify the structures unambiguously.

Therefore we modified the Czech parser `synt` so that it is possible to gain syntactic structures corresponding to the given nonterminal in a number of ways according to the user's choice. To improve the structures detection, we also employed the results of contextual actions used in `synt` as described in Section 4, which increased the precision of morphological analysis by almost 30 %. We also present results of the extraction from sample sentences as well as the usage for shallow valency extraction from annotated corpora.

Petr Sojka, Aleš Horák (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2008, pp. 56–62, 2008.

2 Syntactic parser `synt`

Syntactic parser `synt` [1] has been developed for several years in the Natural Language Processing Centre at Masaryk University. It performs a chart-type syntactic analysis based on the provided context-free head-driven phrase-structure grammar for Czech. For easy maintenance, this grammar is edited in form of a metagrammar (having about 200 rules) from which the full grammar can be automatically derived (having almost 4,000 rules). Contextual phenomena (such as case-number-gender agreement) are covered using the per-rule defined contextual actions.

In recent measures [2, p. 77] it has been shown that `synt` accomplishes a very good recall (above 90 %) but the analysis is highly ambiguous: for some sentences even billions of output syntactic trees can occur. There are two main strategies developed to fight such ambiguity: first, the grammar rules are divided into different priority levels which are used to prune the resulting set of output trees. Second, every grammar rule has a ranking value assigned from which the ranking for the whole tree can be efficiently computed in order to sort the trees on the output accordingly.

For the purpose of the extraction, the internal parsing structure of `synt` is used, the so called *chart*, an acyclic multigraph which is built up during the analysis holding all the resulting trees. What is important about chart is its polynomial size [3, p. 133] implying that it is a structure suitable for further effective processing – as the number of output trees can be up to exponential regarding to the length of the input sentence, processing of each tree separately would be otherwise computationally infeasible. By processing of the chart we refer to the result of the syntactic analysis, i.e. to the state of the chart after the analysis.

3 Extraction of structures

Several ways how to identify the given syntactic structures have been developed respecting the (from the nature of language given) reality that these structures differ a lot in their inner form and thus no universal procedure can be used for all of them. Since we want the output of the extraction to be unambiguous, the extraction covers all possible structures and their combination that result from the analysis. There are two very straightforward approaches for structures detection which consist in extracting the biggest or smallest found structure, however to achieve quality results, more sophisticated methods have to be employed for each structure/nonterminal separately. Speaking about biggest or smallest we mean that regarding to the fact that many of the rules in the grammar used by `synt` are recursive. The results for various nonterminals are listed in Examples 1–4.

- *Example 1.* – clause (nested)
  Input:

Muž, který stojí u cesty, vede kolo.
(A man who stands at the road leads a bike.)
Output:
`[0-9): Muž , , vede kolo` *(a man leads a bike)*
`[2-6): který stojí u cesty` *(who stands at the road)*

- *Example 2.* – verb phrase
  Input:
  Kdybych to byl býval věděl, byl bych sem nechodil.
  (If I had known it, I would not have come here.)
  Output:
  `[0-5)`[1] `: byl býval věděl` *(had known)*
  `[6-10): byl bych nechodil` *(would not have come)*

- *Example 3.* – clause (sequence)
  Input:
  Vidím ženu, která drží růži, která je červená.
  (I see a woman who holds a flower which is red.)
  Output:
  `[0-3): Vidím ženu ,` *(I see a woman)*
  `[3-7): která drží růži ,` *(who holds a flower)*
  `[7-10): která je červená` *(which is red)*
- *Example 4.* – noun phrase
  Input:
  Tyto normy se však odlišují nejen v rámci různých národů a států, ale i v rámci sociálních skupin, a tak považuji dřívější pojetí za dosti široké a nedostačující.
  (But these standards differ not only within the scope of various nations and countries but also within the scope of social groups and hence I consider the former conception to be wide and insufficient.)
  Output:
  `[0-2)`: Tyto normy *(These standards)*
  `[6-12)`: v rámci různých národů a států *(within the scope of various nations and countries)*
  `[15-19)`: v rámci sociálních skupin *(within various social groups)*
  `[23-30)`: dřívější pojetí za dosti široké a nedostačující *(former conception for wide and insufficient)*

4 Morphological refinement

In order to further divide big structures into separate meaningful segments it is possible to part them according to the morphological agreement – i.e. in such a way that words in each structure agree in case, number and gender. To

[1] The numbering denotes a (left inclusive, right exclusive) range of the structure in the input sentence (i.e. words indices).

improve this technique some previously unused results of the syntactic analysis have been involved, namely the contextual actions used by the parser to handle the case-number-gender agreement. In each analysis step, the results of the contextual actions are propagated bottom-up so that they can be used in the next step to prune possible derivations.

Table 1. A comparison of morphological tagging before and after the refinement. The whole sentence in English was: *There was a modern shiny car standing on a beautiful long street.* Note that for readability purpose we abbreviate the tags so that `k7{c4,c6}` stands for `k7c4, k7c6`.

word	before	after
Na *(on)*	k7{c4, c6}	k7c6
krásné *(beautiful)*	k2eA{gFnPc1d1, gFnPc4d1, gFnPc5d1, gFnSc2d1, gFnSc3d1, gFnSc6d1, gInPc1d1, gInPc4d1, gInPc5d1, gInSc1d1wH, gInSc4d1wH, gInSc5d1wH, gMnPc4d1, gMnSc1d1wH, gMnSc5d1wH, gNnSc1d1, gNnSc4d1, gNnSc5d1}	k2eAgFnSc6d1
dlouhé *(long)*	k2eA{gFnPc1d1, gFnPc4d1, gFnPc5d1, gFnSc2d1, gFnSc3d1, gFnSc6d1, gInPc1d1, gInPc4d1, gInPc5d1, gInSc1d1wH, gInSc4d1wH, gInSc5d1wH, gMnPc4d1, gMnSc1d1wH, gMnSc5d1wH, gNnSc1d1, gNnSc4d1, gNnSc5d1}	k2eAgFnSc6d1
ulici *(street)*	k1gFnSc3, k1gFnSc4, k1gFnSc6	k1gFnSc6
stálo *(stand)*	k5eAaImAgNnSaIrD	k5eApNnStMmPaI[2]
moderní *(modern)*	k2eA{gFnPc1d1, gFnPc4d1, gFnPc5d1, gFnSc1d1, gFnSc2d1, gFnSc3d1, gFnSc4d1, gFnSc5d1, gFnSc6d1, gFnSc7d1, gInPc1d1, gInPc4d1, gInPc5d1, gInSc1d1, gInSc4d1, gInSc5d1, gMnPc1d1, gMnPc4d1, gMnPc5d1, gMnSc1d1, gMnSc5d1, gNnPc1d1, gNnPc4d1, gNnPc5d1, gNnSc1d1, gNnSc4d1, gNnSc5d1}	k2eAgNnSc1d1, k2eAgNnSc4d1, k2eAgNnSc5d1
nablýskané *(shiny)*	k2eA{gFnPc1d1rD, gFnPc4d1rD, gFnPc5d1rD, gFnSc2d1rD, gFnSc3d1rD, gFnSc6d1rD, gInPc1d1rD, gInPc4d1rD, gInPc5d1rD, gInSc1d1wHrD, gInSc4d1wHrD, gInSc5d1wHrD, gMnPc4d1rD, gMnSc1d1wHrD, gMnSc5d1wHrD, gNnSc1d1rD, gNnSc4d1rD, gNnSc5d1rD}	k2eAgNnSc1d1, k2eAgNnSc4d1, k2eAgNnSc5d1
auto *(car)*	k1gNnSc1, k1gNnSc4, k1gNnSc5	k1gNnSc1, k1gNnSc4, k1gNnSc5

So far these outcomes in form of morphological values have not been used in any other way. Our enhancement backpropagates these values after the analysis top-down to the chart nodes, i.e. input words, and prunes their original morphological tagging. This leads to more precise morphological analysis and hence it also enables more exact distinction between substructures according to grammar agreement. A detailed example of the impact of morphological refinement on particular sentence is provided in Table 1.

Testing on nearly 30,000 sentences from Czech annotated corpus DESAM [4] has shown that it is possible to increase the number of unambiguously analysed

[2] The inconsistence in tagging on this row has purely technical background – the tag set has been changed.

words by almost 30 % using this method while the number of errors introduced consequently remains very low, as shown in Table 2.

Table 2. Morphological refinement results on the DESAM corpus.

value	before	after
average unambiguous words	20,733 %	46,1172 %
average pruned word tags	38,3716 %	
error rate [3]	< 1,46 %	
number of sentences	29 604	

Parting structures according to their grammatical agreement is useful, for example, when extracting noun or prepositional phrases, as can be seen in Example 5 (compare with Example 4 where the same sentence is extracted without morphological parting).

Example 5.
Input:
Tyto normy se však odlišují nejen v rámci různých národů a států, ale i v rámci sociálních skupin, a tak považuji dřívější pojetí za dosti široké a nedostačující.
(But these standards differ not only within the scope of various nations and countries but also within the scope of social groups and hence I consider the former conception to be wide and insufficient.)
Output:
`[0-4): Tyto normy se však`
(But these standards)
`[6-8): v rámci`
(within the scope)
`[8-12): různých národů a států`
(various nations and countries)
`[13-17): ale i v rámci`
(but also within the scope)
`[17-19): sociálních skupin`
(social groups)
`[23-25): dřívější pojetí`
(former conception)
`[25-30): za dosti široké a nedostačující`
(for wide and insufficient)

Specific modifications how to extract nonterminals with important semantical representation have been developed. Furthermore, these settings can be extended to other (possibly new) nonterminals easily as they are available as command-line parameters.

[3] As an error we consider a situation when the correct tag has been removed during the refinement process. Actually the error rate is even lower since many of the results marked as wrong were caused by an incorrect tag in the corpus.

5 Applications: shallow valency extraction

Currently, a new verb valencies lexicon for Czech, called *Verbalex* [5], is being developed in the NLP Centre. As building of such a lexicon is a very time-consuming long-term task for linguists professionals, it is extremely important to use any possibilities to make this process easier for them. Therefore, we extended the extraction of structures so that it performs a shallow valency extraction from annotated corpora. The main idea is as follows: first, we extract separate clauses, then we identify individual verbs or verb phrases and finally we find noun and prepositional phrases within each clause. Sample output in BRIEF format is provided in Example 6. Moreover, such basic extraction might be used for approximating the coverage of the valency lexicon by finding verbs that are not included there.

Example 6.

```
 ; extracted from sentence: Nenadálou finanční krizi musela
podnikatelka řešit jiným způsobem .
řešit <v>hPTc4,hPTc7
```

*(The businessman had to **solve** the sudden financial crisis in another way.)*

```
; extracted from sentence: Hlavní pomoc ale nacházela v dalších
obchodních aktivitách .
nacházet <v>hPTc4,hPTc6r{v}
```

*(However she **found** the main help in further business activities.)*

```
; extracted from sentence: U výpočetní techniky se pohybuje
v rozmezí od 8000 Kč do 16000 Kč .
pohybovat <v>hPTc2r{u},hPTc6{v}
```

*(By IT [it] **ranges** between 8,000 Kč and 16,000 Kč.)*

6 Conclusions

We presented recent improvements in the Czech parser `synt` that can be used for extracting various syntactic (sub)structures. We also showed practical usage of syntactic analysis for refining morphological tagging as well as examples using the resulting tagging for structures distinction. Furthermore, we presented an application of structures extraction, namely shallow extraction of valencies.

In the future there will be further work on the development of this extraction. We would like to compare the results of morphological refinement with similar oriented methods (e.g. with morphological disambiguation as described in [6]) as well as perform more detailed experiments with the shallow valency extraction on big annotated corpora.

Acknowledgements. This work has been partly supported by the Academy of Sciences of Czech Republic under the project 1ET100300419 and by the Ministry of Education of CR within the Center of basic research LC536 and in the National Research Programme II project 2C06009.

References

1. Kadlec, V., Horák, A.: New meta-grammar constructs in Czech language parser synt. In: Proceedings of TSD 2005, LNCS 3658, Springer-Verlag (2005), pp. 85–92.
2. Kadlec, V.: Syntactic analysis of natural languages based on context-free grammar backbone. Ph.D. thesis, Faculty of Informatics, Masaryk University, Brno (2007).
3. Horák, A.: The Normal Translation Algorithm in Transparent Intensional Logic for Czech. Ph.D. thesis, Faculty of Informatics, Masaryk University, Brno (2001).
4. Pala, K., Rychlý, P., Smrž, P.: DESAM – Annotated Corpus for Czech. In: Proceedings of SOFSEM '97, Springer-Verlag (1997) 523–530.
5. Hlaváčková, D., Horák, A., Kadlec, V.: Exploitation of the Verbalex verb valency lexicon in the syntactic analysis of Czech. In: Proceedings of TSD 2006, Brno, Springer-Verlag (2006), pp. 85–92.
6. Šmerk, P.: Unsupervised learning of rules for morphological disambiguation. In: Proceedings of TSD 2004, Brno, LNCS 3206, Springer-Verlag (2004), pp. 211–216.

Test Suite for the Czech Parser Synt

Vojtěch Kovář and Miloš Jakubíček

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

Abstract. This paper presents a set of tools designed for testing the Czech syntax parser that is being developed at the Natural Language Processing Centre at the Masaryk University, `synt`. Testing the parser against a newly created phrasal tree corpora is very important for future development of the parser and its grammar. The usage of the test suite is not restricted to the `synt` parser but is open to wide scope of applications that provide similar output.

1 Introduction

Automatic syntactic analysis is one of the basic tasks in advanced natural language processing. However, the syntactic analysers (or parsers) developed for the Czech language deal with many serious problems, e.g. low precision or high ambiguity of the parsing results. For this reason, the development of the parsers must continue as effectively as possible and the qualities of the parsers must be continually tested against the corpus data.

This paper concerns a Czech parser `synt` that is being developed at the Natural Language Processing Centre at the Masaryk University (NLP Centre). The parser is based on context-free backbone with additional contextual actions and it features a developed meta-grammar formalism with a fast parsing algorithm. It produces sets of possible derivation phrasal trees and the output can be highly ambiguous. However, a tree-ranking algorithm is implemented that enables the parser to select one "best" tree from the output set in a short time that does not depend on the overall number of trees.

Until recently, there was no larger corpus of phrasal trees available. The only huge treebank for the Czech language was the Prague Dependency Treebank [1] but the dependency formalism is very different from the phrasal one and the conversion between dependency and phrasal structures can produce a large number of errors [2]. At the current time, a new treebank with phrasal trees has been built at the NLP Centre and we plan to use this treebank intensively in the process of the `synt` parser development.

In this paper, we introduce a set of tools (test suite) developed for testing the `synt` parser (as well as any other parser that produces similar outputs) using the new phrasal treebank. We briefly describe both the parser and the treebank and then we characterize the test suite itself: the procedure of testing, used metrics, comparison of a particular test with a reference one and related problems.

Petr Sojka, Aleš Horák (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2008, pp. 63–70, 2008.

2 The `synt` Parser

The `synt` parser is based on a large Czech meta-grammar with context-free backbone and contextual actions. The involved parsing algorithm uses a modification of *head-driven chart parser* [3] that provides very fast parsing even in combination with big grammars. As mentioned in the introduction, the parser output produces set of ranked trees that match the parser meta-grammar.

Besides the parsing algorithm itself, many additional functions are implemented in the system, such as algorithm for finding the best coverage (for sentences that do not match the grammar), efficient selection of N best output trees from the analysis results or using so called *limits*.

The *limits* function is used if the user wants to prune the set of resulting trees according to their structure. The parser gets a set of limits on its input that can look like *0 4 np* and prints only the trees matching all the limits. In the previous example, only the trees would be printed in that a "np" (noun phrase) non-terminal covers the input from position 0 to position 4.

The coverage of the parser grammar is about 92 percent of Czech corpus sentences [4, p. 77]. Its precision was never rigorously evaluated because of insufficient syntactically annotated corpus data. (The only testing against a big corpus data is reported in [2] but the results indicate that the testing data were highly distorted by format conversions.) With the newly created phrasal treebank and test suite, we could make such evaluation. Its results are presented in the Section 4.6.

3 The Brno Phrasal Treebank

The Brno Phrasal Treebank was created in years 2006–2008 as a product of linguist specialists collaborating with the NLP Centre. The corpus contains in overall 86,058 tokens and 6,162 syntactically tagged sentences. The main source of sentences is the Prague Dependency Treebank.

Besides the correct tree in the phrasal formalism, the treebank source files contain information about the source of the text, lemmatized and morphologically tagged format of the text and limits that must be fulfilled by all correct trees. These limits contained in the treebank source files are used in one of the test suite statistics, as explained in following sections.

An example of a treebank sentence is shown in Figure 1.

4 The Test Suite

The test suite is a set of scripts that performs an automatic comparison of the `synt` parser output with the introduced phrasal treebank. Basically, it runs the parser over the morphologically tagged data from the treebank and incrementally computes the statistics according to the parser output.

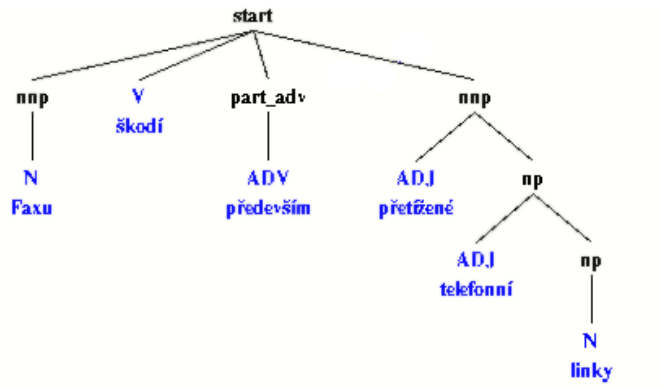


Fig. 1. An example of a treebank sentence

4.1 Included Statistics

The basic statistics we wanted to include in the testing are the following:

- *overall number of parsing trees* – useful for grammar ambiguity estimations.
- *number of limits trees* – or number of trees fulfilling limits. This number tells us how many "correct" trees have been found in the output. Ideally we want only one; if there are no such trees, the output of the parser is incorrect. In case of several trees, the limits recorded in the treebank should be probably more restrictive.
- *similarity* of the parsing results with the correct tree recorded in the treebank.

4.2 Measuring Similarity of Trees

The last of the presented statistics creates two questions:

- What similarity metric to use?
- How to handle ambiguous output of the parser with a tree-to-tree similarity metric?

Our answer to the first question is the usage of the metric called *leaf-ancestor assessment* (LAA) [5] proposed by Geoffrey Sampson in 2000. This metric is considered to be more reliable than the older PARSEVAL metric that is currently used more frequently. We outline the main characteristics of the metric in the following section.

The solution of the second problem is to use three different numbers for evaluation of the ambiguous output of the parser:

- *best tree similarity* – the best score of the LAA similarity metric reached by any tree from the output set.

– *average similarity* – average score of the LAA metric for all trees in the output set.
– *first tree similarity* – score of the best-ranked tree in the output set.

The first number can tell us how good the parser could be if we had an ideal tree-ranking function. The second one predicates of the overall precision of the grammar. The last item is probably the most useful since in most linguistic or NLP applications, we usually want *one best tree* from the parser, not a set; so this is the number that a potential user or advanced NLP application can expect when handling only one tree.

For efficiency reasons, we always take maximum 100 output trees as the whole output set.

Another complication related to the similarity measuring is the fact that the `synt` grammar, especially its set of non-terminals, slightly changes in time. For this reason, we applied *renaming* of the non-terminals in the resulting candidate trees as well as in the treebank trees. Moreover, the renaming of the non-terminals will make testing of other parsers by the same test suite possible and it can fix several small errors in the treebank data as well. The target set of nonterminals is shown in Table 1.

Table 1. Target non-terminals for renaming

nonterminal	description
ABBR	abbreviation
ADJP	adjective phrase
ADVP	adverbial phrase
CLAUSE	clause
CP	conjunctions or punctuation (in the middle of sentence)
ENDS	ending sentence punctuation
NP	noun phrase
PP	prepositional phrase
PREP	preposition
PRON	pronoun
SENTENCE	the whole sentence (without ending punctuation)
VP	verb phrase
TOP	root nonterminal
OTHER	any other constituent (particle, interjection)

4.3 The LAA Parse Evaluation Metric

Every possible parse evaluation metric has to compare two trees – the correct one (also called *gold standard*) and the one output by the parser (also called *candidate*). The LAA metric is based on comparing so called *lineages* of the two trees.

A *lineage* is basically a sequence of non-terminals found on the path from a root of the derivation tree to a particular leaf. For each leaf in the tree, the lineage is extracted from the candidate parse as well as from the gold standard parse. Then, the edit distance of each pair of lineages is measured and a score between 0 and 1 is obtained. The mean similarity of all lineages in the sentence forms the score for the whole analysis. More information about the metric can be found in [5].

In [6], it is argued that the LAA metric is much closer to human intuition about the parse correctness than other metrics, especially PARSEVAL. It is shown that the LAA metric lacks several significant limitations described also in [7], especially it does not penalize wrong bracketing so much and it is not so tightly related to the degree of the structural detail of the parsing results.

In the test suite, we used the implementation of the LAA metric by Derrick Higgins that is available at `http://www.grsampson.net/Resources.html`.

4.4 The output format

The results of each testing are saved in the form of a text file with 6 columns:

- sentence ID
- number of limits trees
- overall number of output derivation trees
- best tree similarity
- average similarity
- first tree similarity

After the test suite completes the whole file, a short summary is printed, as shown in the Figure 2.

```
BASIC TEST RESULTS

No tree in limits       :  1162    sentences
More trees in limits    :  1904    sentences
Not accepted            :   274    sentences
Median number of trees  :    22
Average number of trees :  1400.61
Average LAA (best)      :    91.48
Average LAA (first 100) :    86.28
Average LAA (first)     :    87.79
```

Fig. 2. The summary output of the test suite

4.5 Comparing Two Tests

During the parser development, we usually want to be able to compare several runs of the test suite in order to immediately gain a view of the

impact of changes we have done. This enables us to prevent regressions in the development as well as it makes easier to track the changes history.

Thus, it is possible to perform a test-to-test comparison which outputs a table with test summaries. Furthermore, a detailed lookup of sentence changes is printed so that developers can directly correct any issues (see Figure 3). Currently, we collect following sentence differences (however the system is designed to be easily extended if further details were needed):

- sentences which do not pass the limits anymore
- sentences which newly cause a parser failure/timeout
- sentences with regressions in the number of trees/LAA values.

In order to speed up the comparison even more, an HTML document is produced as well, allowing the user (on-click) to obtain trees to compare after the tree images are created on-the-fly. A view of a tree-to-tree confrontation is provided in Figure 4.

4.6 Evaluation Results and Discussion

In the Figure 2, the results of a real test are shown. We can see that for 1,162 sentences (which is about 20 percent of the treebank) there is no correct tree in the parser output. However, the results of similarity measuring were relatively good – 87.8 percent for the first 100 trees. It can be also seen that the score for these first trees is better than average. This is a strong evidence that the parser ranking algorithm is basically correct. However, it could be still better; with an ideal ranking function we could reach the precision of 91.5 percent.

There is one remaining problem in interpretation of the results. For efficiency reasons, some parsing processes were killed during the testing since they exceeded a fixed time limit. It is an open question how to handle these "killed" sentences. In the evaluation presented above, these sentences were skipped and were not included into the statistic. If we counted them in with a score e.g. 0, the LAA metrics would fall down to 65–70 percent.

5 Conclusions and Future Directions

In the paper, we have presented a newly created test suite for the Czech parser `synt` that uses a new phrasal treebank for the Czech language. We have presented used metrics and procedures needed to get the results as well as outputs useful for developers of the parser. We also presented the precision of the parser measured by the introduced test suite.

In the future development, we mainly want to improve the parser grammar according to the data retrieved from the testing suite. At the same time, we plan to enhance the test suite according to the feedback we will get from the developers of the parser.

```
Summary
================================================================================
|test name           |test-2008-10-30-18-4-58|test-2008-10-30-11-6-29|  diff  |
--------------------------------------------------------------------------------
|sentences           |          6162         |          6162         |   ==   |
|passed limits       |      4551 (74 %)      |      4552 (74 %)      |   ++   |
|failed              |           0           |           0           |   ==   |
|timed out           |          1611         |          1610         |   ++   |
|more than one tree  |      1904 (31 %)      |      1904 (31 %)      |   ==   |
|median trees count  |          72           |          72           |   ==   |
|LAA Best            |        0.5667         |        0.5667         |   ==   |
|LAA Avg             |        0.5349         |        0.5349         |   ==   |
|LAA First           |        0.5442         |        0.5442         |   ==   |
================================================================================
Details:
===========
Sentences which do not pass the limits anymore:
[]
Sentences which newly cause a failure:
[]
Sentences which have been newly timed out:
[]
Sentences with more trees than previously:
[5708]
Sentences with lower LAABest than previously:
[]
Sentences with lower LAAAvg than previously:
[]
Sentences with lower LAAFirst than previously:
[]
```

Fig. 3. A test-to-test comparison output (random tests)

Acknowledgements. This work has been partly supported by the Academy of Sciences of Czech Republic under the projects 1ET100300414 and 1ET100300419 and by the Ministry of Education of CR in the National Research Programme II project 2C06009.

References

1. Hajič, J.: Building a syntactically annotated corpus: The Prague Dependency Treebank. In: Issues of Valency and Meaning, Prague, Karolinum (1998) 106–132.
2. Horák, A., Holan, T., Kadlec, V., Kovář, V.: Dependency and Phrasal Parsers of the Czech Language: A Comparison. In: Proceedings of the 10th International Conference on Text, Speech and Dialogue, Pilsen, Czech Republic, Springer Verlag (2007) 76–84.
3. Horák, A., Kadlec, V., Smrž, P.: Enhancing best analysis selection and parser comparison. In: Lecture Notes in Artificial Intelligence, Proceedings of TSD 2002, Brno, Czech Republic, Springer Verlag (2002) 461–467.

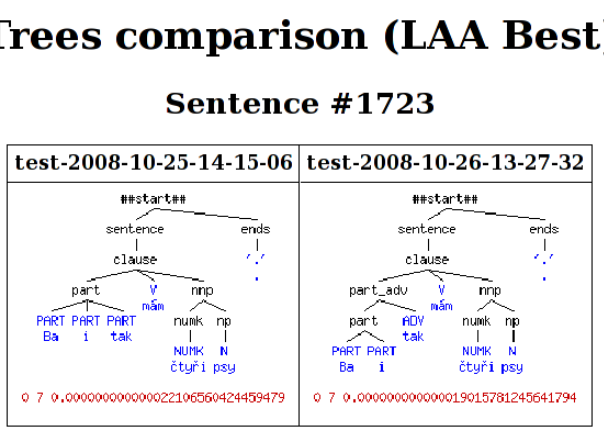


Fig. 4. A tree-to-tree confrontation selected in the HTML test-to-test comparison.

4. Kadlec, V.: Syntactic analysis of natural languages based on context-free grammar backbone. Ph.D. thesis, Faculty of Informatics, Masaryk University, Brno (2007).
5. Sampson, G.: A Proposal for Improving the Measurement of Parse Accuracy. International Journal of Corpus Linguistics **5**(01) (2000) 53–68.
6. Sampson, G., Babarczy, A.: A test of the leaf-ancestor metric for parse accuracy. Natural Language Engineering **9**(04) (2003) 365–380.
7. Bangalore, S., Sarkar, A., Doran, C., Hockey, B.A.: Grammar & parser evaluation in the XTAG project (1998)
`http://www.cs.sfu.ca/~anoop/papers/pdf/eval-final.pdf`.

Computing Idioms Frequency in Text Corpora

Jan Bušta

Faculty of Informatics, Masaryk University, Brno, Czech Republic
xbusta@fi.muni.cz

Abstract. The idioms are phrases which meaning is not composed from the meanings of each word in the phrase. This is one of the natural examples of violating the principle of compositionality that means that idioms are in area of natural language processing problem of meaning mining. To count the frequency of phrases such idioms in corpora has one big aim: To get to know which phrases we use often and which less. We do it to be able to start with getting the meaning of the whole phrases not just each word. This improves the understanding natural language.

1 Idioms

First step how be able to count the idiom frequency is to recognize the idiom. Although we have a dictionary (Slovník české frazeologie a idiomatiky), the idioms are in corpus in non-standard form. The question, what is and what is no more an idiom, is quite difficult to define. The word order in idioms is not fixed, there is for example an idiom *hrát druhé housle*[1] you find it in this form in the dictionary, but in the real text, the idiom could be *. . . já druhé housle hrát nehodlám*. It is shown that the word order is switched, the noun phrase is in the front of the verb phrase.

The structure of idioms in the dictionary are well primary divided in two categories:

- **the verb-based idioms and phrases** – the main word is a verb and the other words, which are optional, create only the context or/and further qualification,

  Example 1. hnout / pohnout někomu žlučí, dostat zaslouženou odměnu, být odkázán / bejt vodkázanej sám na sebe

- **the non-verb-based idioms and phrases** – the main meaning depends on the non-verbal word, very often on nouns (substantives), or is composed form the many words which are on the same meaning level.

  Example 2. bída s nouzí, brána pekla / do pekla / do pekel, bez velkých / dlouhých / zbytečných ceremonií

Note 1. Whether the phrase is verbal- or non-verbal-based depends on the language, the same meaning could be said in different forms in each language.

[1] All idioms in text are from SČFI [1]

Petr Sojka, Aleš Horák (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2008, pp. 71–74, 2008.

This division is very useful because the approach to this two groups is totally different. While for the verb-based idioms should the query take into account possible noun- and verb-phrase switching, by the non-verb-based idioms is the situation easier, because the word switching in this case is not common (there are some exception, but the idioms created this way sounds archaic, e. g. *země dary*).

Changing the case, gender or tense in idioms is not a real computing problem. Some idioms can occur in a specific positional form only, the phrase has an idiomatic meaning only if the words are next to each other, usually is that an adjective and a substantive which specify the base word, but in some idioms is also not fixed the number of words. You can start with the first part (whatever the first part is), insert some words which can specify the meaning, and finish the original phrase (see next example). This makes some problems with selecting the idiom in text.

Example 3. . . . , že by měla v budoucnu ***hrát*** *ve svém regionu* ***druhé housle****. . .*

1.1 Frege's principle

Idioms are the phrases which violating Frege's principle of compositionality. This fact makes this work meaningful, because we are not able to translate (or get the sense) from the sentence if it contains an idiom; first we have to define the meaning of the parts in sentence thereafter is the clear road to processing. Idioms can not be processed as it is but have to be preprocessed. The easiest way it to give them the fixed meaning/translation. It can be hard to work with all idioms therefore we will select the most frequented idioms.

2 Corpora

The primary corpus data come form the SYN2000c corpus made by Institute of the Czech National Corpus at Charles University in Prague. This corpus includes more than 100 millions words from complete texts sources. The SYN2000c is a synchronous corpus, which means that most the documents added into it have been published in the same time (1990–1999). The SYN200c corpus contains also some older documents from authors which were born after 1880. [2]

Using this corpus provides very good overview on the Czech language, the solution corresponds with today's language. Another advantage is that this corpus is tagged therefore is no problem with forms of words in any case, gender or tense.

3 Querying

The main work is in the querying. How to query the corpus if it contains the given idiom or not and if yes how many times. There is no problem, to get

"some result", but it happens, that we have to think about the idiom structure and adapt the query to the idiom we are looking.

For querying the corpus is the Corpus Query Language[3] which gives the power to create complex queries. It allows to specify the context or distance of each words in idioms. The result reflects very sensitively the used query.

Next example show how search the verb-based idiom *hrát druhé housle* in corpus.

Example 4.
```
[lemma="hrát"][word!="\."]{0,5}"druhé""housle"|
"druhé""housle"[word!="\."]{0,5}[lemma="hrát"]
```

The result of this query will be a list of phrases which begins with any from of the first part (in this this case the verb *hrát*, next word can be everything else except the dot sign, there should be 0–5 words this type (inserted words) and at the and is the other part of idiom which are consist from the two contiguous words *druhé* and *housle*. The other part implements the switching of the parts of the idiom. The count of found idioms using this query is 47, but if we searching only the occurrence of the word phrase *druhé housle*, the count will be 52. The difference between this two results is in the fact, that the phrase *druhé housle* can be used separately in the non-idiomatic meaning (music stuff). Knowing the right borders of idiom will decide, if the phrase is an idiom or not.

The situation in the field of non-verb-based idioms seems to be easier, but there are other things which could be solved. Many of non-verb-based idioms have more than one. The structure of them can consist from the static part and the part which could be changed. This second part is created from word (or words) which have the same/near meaning. The idioms *dar řeči*, *dar jazyka* and *dar výmluvnosti* are according to the dictionary the same. This idioms should be detect and searched as:

Example 5.
```
[lemma="dar"][word="řeči|jazyka|výmluvnosti"]
```

In this idiom example is also impossible to divide the idiom in two parts, this is the property of majority of non-verb-based idioms.

A special group of idioms are one-word idioms, e. g. *žízeň*. In this case is the frequency of idiom identical to the plain frequency of the word which is not exact. Many of occurrences are the words in his base meaning, to divide the base and the idiomatic meaning of the word is context dependent. There is no solution how to recognize this idioms without any other supporting method.

In the idioms written in SČFI are sometimes a word (words) of idiom in non-literary form. It would not be if the *lemma* will match the *lemma* of the literary equivalent, but there are not, it makes difficulties by searching in corpus.

3.1 Dividing the idioms

To find an automatic procedure of making the queries is the good classifying the idioms and do more specific groups of idioms than verb- and non-verb-based

in which would be such of them which satisfy the prepared slots for making the final query.

It seems to by useful (in case of verb-based idioms) to divide them by their possibility to change the position of parts of the idioms, by possibility to accept the inserted words in the middle, by the alternatives parts (synonymic phrases inside the idiom). One of the important variable, which should be set, is the maximal distance of the parts of idioms. If this value will be to slow, some idioms will be not found; if it will be to high, some other phrases will be found which will be not idioms.

Non-verb-based idioms are easier to recognize also the dividing in groups linked with their structure. For the non-verb-based idiom groups is the main characteristic if there is a alternative part or words have to be side-by-side.

4 Conclusion

The processing the idioms, cleaning the data and preparing the form of the idioms is the fist step to be able to create the queries which will match the highest count of the idioms without matching any other phrases which are in the corpus but are not idioms.

The second step is prepare the skeleton of query, the slots have to be as much as possible specific (to accept only the right group of idioms).

After doing the previous steps we can start with the querying the corpus. The results are only very hard to evaluate in particular because there are no results done. We can sure compare results of some idiomatic searches but not all results.

Maybe will be some methods used also by computing the frequency of idioms in other language although the structure of idioms is language dependent.

Acknowledgments. This work has been partly supported by the Academy of Sciences of Czech Republic under the projects 1ET100300419 and 1ET200610406, by the Ministry of Education of CR within the Center of basic research LC536 and in the National Research Programme II project 2C06009 and by the Czech Science Foundation under the project 407/07/0679.

References

1. Čermák, F., collective: Slovník české frazeologie a idiomatiky. Academia (1994).
2. Institute of the Czech National Corpus: Structure of technical and other specialised literature according to the technical orientation (2008) [Online; accessed 12-November-2008].
3. Christ, O.: A modular and flexible architecture for an integrated corpus query system. Technical report, Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart (1994).

Plagiarism Detection through Vector Space Models Applied to a Digital Library

Radim Řehůřek

Faculty of Informatics, Masaryk University
xrehurek@fi.muni.cz

Abstract. Plagiarism is an increasing problem in the digital world. The sheer amount of digital data calls for automation of plagirism discovery. In this paper we evaluate an Information Retrieval approach of dealing with plagiarism through Vector Spaces. This will allow us to detect similarities that are not result of naive copy&paste. We also consider the extension of Vector Spaces where input documents are analyzed for term co-occurence, allowing us to introduce some semantics into our approach beyond mere word matching. The approach is evaluated on a real-world collection of mathematical documents as part of the DML-CZ project.

1 Introduction

1.1 What is plagiarism?

With the advances in technology (storage and processor performance, database and scanning systems, user interfaces), creating large digital collections of documents becomes largely an engineering task. *Digital library* is a centrally managed digital collection of documents, such as texts, multimedia images, music or videos. At the same time, the electronic medium makes it easier than ever to plagiarize accessible documents, or portions of them. This discourages information providers from publishing their content, in turn crippling the digital libraries. The idea of stealing someone's work is of course not new, but digital technology and the Internet make reproduction and distribution of documents much faster, easier and safer than the tedious paper or CD-based methods of the past.

According to the Merriam-Webster Online Dictionary [1], to "plagiarize" means

1. to steal and pass off (the ideas or words of another) as one's own
2. use (another's production) without crediting the source
3. to commit literary theft
4. to present as new and original an idea or product derived from an existing source.

Petr Sojka, Aleš Horák (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2008, pp. 75–83, 2008.

1.2 Why plagiarism detection?

One salient area where plagiarism becomes a major problem is the education system. The problem of students turning to the Internet for a quick-fix homework solution which shortcuts around the time-consuming work of writing programming assignments and research papers is becoming very serious. This area has even been noted by the commercial sector already. A number of paper mill services exist, offering plagiarized papers to students, sometimes even with the added option of having the paper "customized". According to a 2002 study by McCabe [2], 10% of American college students have partially copied their assignment from the Internet without proper citation, with 5% turning in verbatim copies from web pages and term-paper mills. The situation is even worse for high school students, with the figures at 52% and 16% respectively. The numbers are rising every year, not the least because plagiarizing is becoming a common (if not acceptable) part of our educational culture. The reward for doing the hard work yourself is mostly moral, and detection and punishment of plagiators very rare.

Also notable is connection between detecting plagiarism in programming and natural languages. The latter are inherently more difficult, because a plagiarized program must, in order to retain the same semantics, also retain very similar syntax. A similar syntactical parse tree of a program is thus in itself highly indicative of plagiarism, something not true for natural languages, where the connection between syntax and semantics is much more variable and vague.

Plagiarism detection as considered in this paper is a computational means of detecting the above mentioned plagiarism violations. As such it includes copy detection, which is the most straightforward case of plagiarism that duplicates parts of documents verbatim. Copy detection is not necessarily useful strictly for unlawful violations; a possible scenario is one where user is activelly sifting through documents from a particular domain. Here the 'original', or *registered* documents are simply documents that have been seen already, and the user is likely not interested in minor modifications (retransmitted or forwarded messages, different versions or editions of the same work, documents coming from mirror sites and so on). He aims to gather topically related documents, without any explicit regard to plagiarism. This is a task studied in the field of Information Retrieval (IR), and indeed in general terms plagiarism detection in digital libraries can be seen as an instance of IR.

2 Document Representation

Vector Space Model (VSM)

Information Retrieval is the part of computer science concerned with retrieving, indexing and structuring digital objects (e.g, text documents, images, videos) from collections (e.g., the Web, corpora). Although several different models have been proposed (see e.g. [3]), the one relevant for this section is the

Vector Space (VS) Model. Here objects are not represented directly, but rather approximated by *features*. What constitutes a feature is application dependent – in our case of text retrieval, most common choice are terms as delimited by white space, or term bigrams. These features are then assigned specific values for each object, leading to a representation of the object by a vector. Even though assignment of discrete values (e.g., $0, 1$ or $0, 1, \ldots, n$) is possible, most extensions to the basic model modify the values by weights, making the vector real-valued. Documents proximity can then be estimated by vector similarity measures, such as vector dot product, cosine similarity and others.

The VS model implies several choices: firstly, which features to extract from the documents, secondly, what weights to assign them and finally how to compute document similarity. The standard practise is to take the set of all tokens that occur in the document and note their frequencies. This tacitly assumes position independence within the document, and also independence of terms with respect to each other. This assumption is intuitivelly wrong (the term 'surf' has different meaning within the context of surfing on the Web and surfing on the beach), but empirical NLP studies nevertheless report good results using it. This approximation of a document by a set of its terms (and their frequencies) is called the *Bag of Words (BOW)* approximation.

Latent Semantic Indexing

To overcome the limitations of simple term overlap, semantic modifications of VS were introduced. One of them is Latent Semantic Indexing (LSI), a technique based on Vector Space model which aims to create associations between conceptually connected documents and terms. Research into LSI originated with [4]. LSI uses linear algebra techniques (i.e., Singular Value Decomposition, SVD), as explained in [5]. The following paragraphs give brief introduction into theoretical background and intuition into how LSI operates on textual domain. A very enticing feature of LSI is that it is a so-called unsupervised method, meaning that no explicit input of knowledge is required for training. It has been shown that LSI has good retrieval performance [10].

Let m be the rank of a term-document matrix M, which may be the TF·IDF matrix described in the previous IR section. We may decompose M into $M = U \cdot S \cdot V^T$, where U (size $t \times m$) and V^T (size $m \times d$) have orthonormal columns and S is diagonal. The columns of U (resp. V) are called the left (resp. right) *singular vectors* (or *eigenvectors*) and are the (normalized) eigenvectors of $M \cdot M^T$ (resp. $M^T \cdot M$). The values in S are the (positive) square roots of the eigenvalues of $M \cdot M^T$ (or equivalently, of $M^T \cdot M$). They are positive real numbers, because $M \cdot M^T$ is symmetric and positive definite. The decomposition process is called *Singular Value Decomposition (SVD)*[1]. Without loss of generality, we can assume the positive real diagonal elements in S, called *singular values*, are sorted by their magnitude, and the corresponding *left and right eigenvectors* in U, V^T are

[1] The Singular Value Decomposition is used to solve many problems (e.g. pseudo-inverse of matrices, data compression, noise filtering) and is a least squares method. LSI uses it to find a low rank approximation of the term-document matrix M.

transposed accordingly. By keeping only the k largest singular values we can reduce S to S_k of size k . Similarly if we keep only the first k columns of U and first k rows of V^T, we get $M_k = U_k \cdot S_k \cdot V_k^T$ (with dimensionalities of $(t \cdot d) = (t \cdot k) \times (k \cdot k) \times (k \cdot d)$). This process is depicted in Figure 1. We call M_k the *rank-k approximation* of M and k the *number of factors*. In fact, as shown in [6], the Eckart-Young theorem states that M_k is the best rank-k approximation of M with respect to the Frobenius norm (2-norm for matrices). How to select the optimal number of latent dimensions k is still an open problem. However empirical studies show that values between 100 and 300 result in best text retrieval performance. In [7,8] the authors propose a statistical test for choosing the optimal number of dimensions for a given collection.

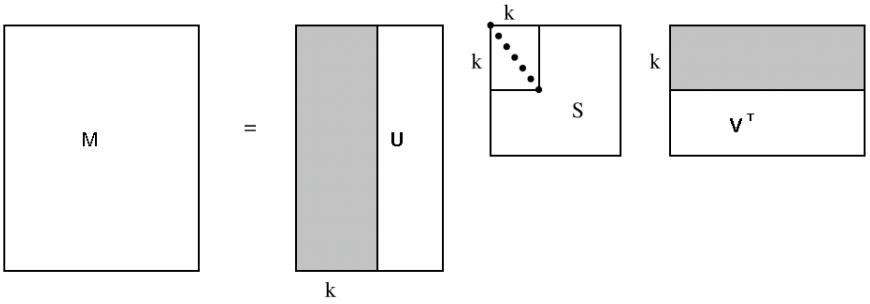


Fig. 1. LSI concept formation: Rank-k matrix approximation of M is obtained by truncating the U, S and V^T matrices from Singular Value Decomposition. Figure taken from [5].

3 DML-CZ

Czech Digital Mathematics Library (DML-CZ) [9] is a project aiming to collect historical mathematical papers and documents within the domain of Bohemia. This includes scanning and using OCR on old papers from pre-digital era. All documents are carefully processed, enriched with metadata and made accessible via web tool called Metadata editor `editor.dml.cz`. The collection, with an additional input of 15,767 articles from NUMDAM, contains 21,431 relevant articles. Out of these, there are 8,145 articles in English suitable for our experiments.

Having such collection offers interesting challenges – in what way do we let our users browse the library? All mathematical articles are reviewed, plus the group of interested people is rather narrow, so plagiarism is unlikely. But still the questions can be asked – are there any suspiciously similar documents within our library? Can document similarity facilitate and enhance browsing experience of the user?

To apply our VSM method as described above, we converted the 8,145 articles to vectors, using both TF·IDF and LSI. For LSI, we reduced dimensionality to the top 200 concepts, in accordance with common IR practice. Then we evaluated pairwise document similarity, using angle distance (cosine measure, similarity range is $\langle 0.0, 1.0 \rangle$) and also plotted the results as 2D matrices. An example of a (part of) similarity matrix for LSI is in Figure 2.

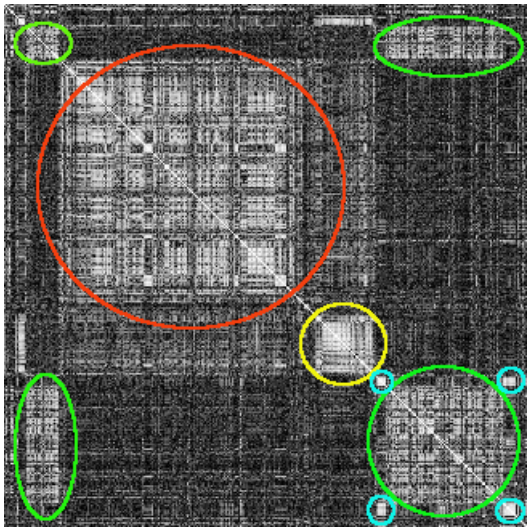


Fig. 2. An example of pair-wise document similarity on a subset of documents. Each pixel represents similarity of one pair of documents, the whiter the more similar. Note that the diagonal is necessarily white, because a document is always maximally similar to itself. The method used is Latent Semantic Indexing. See text for information on the highlighted regions.

4 Results

First question to answer is how much do TF·IDF and LSI differ on our dataset. Statistics show that the mean difference over all articles is 0.0299, with standard deviation of 0.037. Inspection reveals that in most cases the scores are indeed very similar, but there are also many pairs of documents for which the two methods vary widely, as the comparison of mean and standard deviation would suggest. See Appendix for an example of a pair of documents where TF·IDF suggested no similarity (score of 0.08) while LSI scored 0.98.

Perhaps more interesting than the numbers themselves is how well does this vector similarity translate to similarity as perceived by users. Unfortunatelly we do not have a referential tagged corpus of pair-wise document similarities

to compare our results against. However, thanks to the nature of our dataset, we have access to article metadata. One piece of metadata present for each of our articles is its position within MSC classification [11] hierarchy. This is a fixed taxonomy of mathematical areas to which documents are manually assigned by the author or the reviewer. In Figure 2, we selected one node in the MSC hierarchy and considered only those documents in our collection that fall into this category. The category is named *20: Group theory and generalizations* and is further subdivided into smaller categories (*20Dxx Abstract finite groups* etc.). We group documents along the axes according to these subcategories and observe how well does the suggested similarity – represented by shade of gray – correspond to subcategory clusters suggested by MSC. Although there are similarities between individual articles all over the graph, we may observe there are four main “light" clusters. These are highlighted in red, yellow, green, blue and correspond to articles from categories 20Dxx+20Exx+20Fxx, 20.30+20Kxx, 20.92+20.93+20Mxx and 20Lxx+20Nxx, respectively. Descriptions of these ten subcategories of *Group theory and generalizations* are:

- 20.30 (1959-1972) Abelian groups
- 20.92 (1959-1972) Semigroups, general theory
- 20.93 (1959-1972) Semigroups, structure and classification
- 20Dxx Abstract finite groups
- 20Exx Structure and classification of infinite or finite groups
- 20Fxx Special aspects of infinite or finite groups
- 20Kxx Abelian groups
- 20L05 Groupoids (i.e. small categories in which all morphisms are isomorphisms). For sets with a single binary operation, see 20N02; for topological groupoids, see 22A22, 58H05.
- 20Mxx Semigroups
- 20Nxx Other generalizations of groups

Note that all of the suggested clusters are meaningful and also that the algorithm correctly linked obsolete categories 20.92 and 20.93 (used between the years of 1959 and 1972) with their new version of 20Mxx. Although these visual results cannot substitute full analytical evaluation, they are nevertheless quite encouraging.

Next step is to analyze highly similar documents for plagiarism. As mentioned above, finding actual plagiates is highly unlikely due to the nature of the domain. Indeed, analysis shows that all suspicious documents are in fact conference announcements, in memoriams and the like. If there was plagiarism present in the dataset, its complexity was beyond both LSI’s and the author’s ability to detect it.

5 Conclusion

We have presented a robust statistical method for text similarity, applied to a collection of real documents. These documents come from a digital library

of mathematical texts and also have metadata attached, which allowed us to visually compare quality of document similarity. Although our application of plagiarism detection did not yield any positive hits, it nonetheless serves as proof of concept and can be extended and used on other collections.

Acknowledgements. This study has been partially supported by the grants 1ET200190513 and 1ET100300419 of the Academy of Sciences of the Czech Republic and 2C06009 and LC536 of MŠMT ČR.

References

1. Merriam-Webster Online Dictionary, November 2008, `http://www.m-w.com/dictionary/plagiarize`.
2. McCabe, D., 2002. Cheating: Why Students Do It and How We Can Help Them Stop. American Educator (2001–2002) winter, pages 38–43.
3. Regis Newo Kenmogne: Understanding LSI via the Truncated Term-term Matrix. Diploma Thesis at the University of Saarland, Dept. of Computer Science, May 2005.
4. Furnas, Deerwester, Dumais, Landauer, Harshman, Streeter and Lochbaum, 1988. Information retrieval using a singular value decomposition model of latent semantic structure. In: *Proceedings of the 11th annual international ACM SIGIR conference on Research and development in Information Retrieval*, pages 465–480.
5. Michael W. Berry, Susan T. Dumais, and Gavin W. O'Brien. Using linear algebra for intelligent information retrieval. SIAM Review, pages 573–595, December 1994.
6. G. H. Golub, C. F. Van Loan, 1989. Matrix Computations. John Hopkins Press.
7. Zha, H., Simon, H., 1998. A subspace-based model for latent semantic indexing in information retrieval. In *Proceedings of the Thirteenth Symposium on the Interface*. pp. 315–320.
8. Ding, C. H. Q., 1999. A similarity-based probability model for latent semantic indexing. In *Proceedings of the Twentysecond Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pp. 59–65.
9. Bartošek, M. and Lhoták, M. and Rákosník, J., Sojka, P. and Šárfy, M., 2008. DML-CZ: The Objectives and the First Steps. In *CMDE 2006: Communicating Mathematics in the Digital Era*, AK Peters Ltd., pages 69–79.
10. Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., Harshman, R. A., 1990. Indexing by latent semantic analysis. *Journal of the American Society of Information Science* 41(6), pages 391–407.
11. The Mathematics Subject Classification (MSC) taxonomy, November 2008, URL `http://www.ams.org/msc/`

Appendix: TF·IDF vs. LSI Differences

Below there are two articles mentioned in the text where TF·IDF and LSI scores differ dramatically.

On an unrelated note, observe the multiple OCR errors present in the text. These types of low level (character level) errors render application of more refined, semantic-based methods of text analysis very problematic. One of the

advantages of more crude, statistical methods such those based on VSM used in this paper is that these errors are not complete show-stoppers for plagiarism detection.

1. **Czechoslovak Mathematical Journal, vol. 24 (99) 1974, Praha**
   NEWS and NOTICES IN MEMORIAM PROF. RNDr. KAREL CERNY On 15 January 1974, RNDr. Karel Cerny, Associated Professor of Mathematics at the Czech Technical University, died in Prague. Prof. Cerny was born on 6 July 1909 at Zbyslavice near Caslav. After completing his mathe- matical studies at Charles University in 1933 he became lecturer at the Faculty of Mechanical Engineering. He remained member of staff of the Faculty till 1953 except for the years 1942-45 when he suffered from Nazi persecution. In 1953 he was appointed Associated Professor (Dozent) first at the Faculty of Architecture and later at the Faculty of Civil Engineering of the Czech Technical University. Prof. Cerny spared no effort in his educational activity which may be characterized by his full devotion and responsible approach. The scientific interest of K. Cerny, who had been a pupil of Prof. V. Jarnik, was concentrated on the theory of numbers, particularly on the metric theory of diophantine approximations. A more detailed biography of Prof. Cerny is published in Cas. pest. mat. 99 (1974), 321 - 323. Editorial Board
2. **ARCHIVUM MATHEMATICUM (BRNO) Vol. 26, No. 2-3 (1990), 65-66**
   THIS ISSUE OF ARCHIVUM MATHEMATICUM IS DEDICATED TO THE NONAGENERIAN OF * ACADEMICIAN OTAKAR BORtFVKA Academician Otakar Boruvka, Nestor and legend of the Brno mathematicians, long ago one of the leaders of the Czechoslovak mathematical life, a prominent representative of our science abroad, excellent teacher and oiganizer of the scientific life was ninety on May 10, 1989. In full mental freshness, creating activity, in enviable spirit, in constant interest in mathematical events. In 1920-as a student-he passed from the Czech Technical University to the newly founded Faculty of Science of the Brno University and here he passed a state examination in mathematics and physics in 1922. From the year 1921he was a lecturer in the year 1928 he became an associate professor, from the year 1934 he was a professor assistant and'from the year 1946 (with the effectivness from the year 1940) he was a regular professoi of our faculty. From the year 1970 he is a member of the Mathematical Institute of the Czechoslovak Academy of Sciences' in Brno. For the time being he is an author of 84 original scientific papers from the area of differential geometry, general algebra and differential equations and 50 further popular and bibliografical papers. For his results he was awarded a State Prize of Klement Gottwald in the year 1959 and Order of Labour in the year 1965, ⟨hr id="0072"/⟩ from the year 1953 he was a corresponding member and from the year 1965 a regular member of the

Czechoslovak Academy of Sciences, he is an honourable doctor of the Komensky University in Bratislava, and honourable member of the Association of the Czechoslovak Mathematicians and Physicists and he received a number of medals and diplomas of the universities and scientific associations in our country and abroad. Last but not least, he gave rise to this journal (25 yeai ago, in 1965) and was its first editor-in-chief. The rare life anniversary of the Academician Otakar Boruvka is of course associated with a numbei of summary publications in professional and popular press (e.g. Czech. Math. Journal, vol. 39 (113) 198?, 382-384). To us, belonging to the generations of his students, members of scientific seminars, founded or oriented by him, to those, inspired by his work, to his younger collaborators and colleagues and to those esteeming his character, is, however, this reality not only a reason for valorizing his admirable work but also for an oportunity to express our homage to our honoured person by the results of our works. We wish to Academician Boruvka health and good humour in ordei to be able to give away, in further years, from the treasury of his wisdom and experience. Photo: J. France

Automatic Web Page Classification

Jiří Materna

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

Abstract. Aim of this paper is to describe a method of automatic web page classification to semantic domains and its evaluation. The classification method exploits machine learning algorithms and several morphological as well as semantical text processing tools. In contrast to general text document classification, in the web document classification there are often problems with short web pages. In this paper we proposed two approaches to eliminate the lack of information. In the first one we consider a wider context of a web page. That means we analyze web pages referenced from the investigated page. The second approach is based on sophisticated term clustering by their similar grammatical context. This is done using statistic corpora tool the Sketch Engine.

Key words: automatic classification, machine learning, web document, thesaurus

1 Introduction

1.1 Motivation

At the present time the World Wide Web is the largest repository of hypertext documents and is still rapidly growing up. The Web comprises billions of documents, authored by millions of diverse people and edited by no one in particular. When we are looking for some information on the Web, going through all documents is impossible so we have to use tools which provide us relevant information only. The widely used method is to search for information by fulltext search engines like Google[1] or Seznam[2]. These systems process list of keywords entered by users and look for the most relevant indexed web pages using several ranking methods. Another way of accessing web pages is through catalogs like Dmoz[3] or Seznam[4]. These catalogs consist of thousands web pages arranged by their semantic content. This classification is usually done manually or partly supported by computers. It is evident that building large catalogs requires a lot of human effort and fully automated classification

[1] http://www.google.com [2] http://search.seznam.cz [3] http://www.dmoz.org
[4] http://www.seznam.cz

Petr Sojka, Aleš Horák (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2008, pp. 84–93, 2008.

systems are needed. However several systems for English written documents were developed (e.g. [1,2,3,4,5]) the approaches do not place emphasis on short documents nor on the Czech language.

1.2 Objective

Classical methods of text document classification are not appropriate for web document classification. Many of documents on the Web are to short or suffer from a lack of linguistic data. This work treats with this problem in two novel approaches:

- Experiments have proved that hypertext links in web documents usually direct to documents with similar semantic content. This observation leads to use these referenced web pages as an extension of the investigated one for the purposes of processing their linguistic data as well. However there are some restrictions. The referenced documents must be placed on the same server (to avoid joining advertisement or other non-related material) and a level of recursion must be limited. We experimentally set the limit to 2.
- The former method increases amount of linguistic data for the most part of documents enough but there is another problem. To use machine learning algorithms we need to build a high dimensional vector space where each dimension represents one word from or phrase. In spite of the fact that several machine learning algorithms are adjusted to high number of dimensions, in this case the high number of dimensions decreases algorithm accuracy and we have to proceed to dimensional clustering. The joining of two or more dimensions (in this case words) is based on using a special thesaurus built on training data. The method will be described more precisely in the Section *Term clustering*.

2 Preprocessing

In order to use machine learning algorithms we need to build a training data set. There were selected 11 domains (*Cestování, Erotika, Hry, Informační a inzertní servery, Kultura a umění, Lidé a společnost, Počítače a internet, Sport, Věda a technika, Volný čas a zábava, Zpravodajství*) according to the top-level domains in `http://odkazy.seznam.cz` catalog and for each domain collected 1 GB of sample data.

2.1 Data Cleaning

Despite of selecting restricted document content-types (HTML, XHTML) it is necessary to remove noise from the documents. An example of unwanted data is presence of JavaScript (or other scripting languages) as well as Cascading Style Sheets (CSS) and the most of meta tags. Elimination of such data was mostly done by removing *head* part of the document (except of content of

the *title* tag which can hold an important information about domain). As other unwanted data were marked all n-grams (n>10) where portion of non alphanumeric characters was grater than 50 %.

Very important issue of document preprocessing is charset encoding detection. However the charset is usually defined in the header of the document, it is not a rule. We have used a method of automatic charset detection based on byte distribution in the text [6]. This method works with a precision of about 99 %.

A lot of web sites allows user to chose language. Even some web pages on the Czech internet are primarily written in foreign language (typically in Slovak). With respect to used linguistic techniques, we are made to remove such documents from the corpus. The detection of foreign languages is similar to charset encoding detection based on typical 3-gram character distribution. There has been built a training set of Czech written documents and computed the typical distribution. Similarity of training data with the investigated documents is evaluated using cosine measure.

2.2 Corpus construction

Cleaned raw data serve as a groundwork for the training corpus construction. To represent corpus data we use vertical text with following attributes:

- **word** – original word form,
- **lemma** – the canonical form of a word. To get lemma we have used Ajka tagger [7] and disambiguator Desamb [8],
- **tag** – morphological tag of a word (obtained from Ajka).

To process data has been used corpus manager Manatee [9] which offer many statistical functions as well as the Sketch Engine tool [10]. This system can extract so called word sketches which provide information about usual grammatical context of terms in corpus and are used for the thesaurus construction.

3 Document Model

In order to use these data in machine learning algorithms we need to convert them into appropriate document models. The most common approach is vector document model where each dimension of vector represents one word (or token in corpus). There are several methods of representing the words.

Let m is number of documents in the training data set, $f_d(t)$ frequency of term t in document d for $d \in \{1, 2, \ldots, m\}$ and *Terms* set of terms $\{t_1, t_2, \ldots, t_n\}$.

3.1 Binary representation

Document d is represented as a vector $(v_1, v_2, \ldots, v_n) \in \{0, 1\}^n$, where

$$v_i = \begin{cases} 1 & \text{if } f_d(t_i) > 0 \\ 0 & \text{else} \end{cases}$$

3.2 Term frequency representation

Document d is represented as a vector $(v_1, v_2, \ldots, v_n) \in \mathbb{R}^n$, where

$$v_i = \frac{f_d(t_i)}{m}$$

3.3 Term Frequency – Inverse Document Frequency (TF-IDF)

Disadvantage of previous two methods may be a fact of treating with all terms in the same way – they are not weighted. This problem can be solved by using IDF coefficient which is defined for all $t_i \in Terms$ as:

$$IDF(t_i) = log_2 \left(\frac{m}{|\{j : f_j(t_i) > 0\}|} \right)$$

By combining TF and IDF we get:

$$v_i = \frac{f_d(t_i)}{m} \cdot log_2 \left(\frac{m}{|\{j : f_j(t_i) > 0\}|} \right)$$

For TF and TF-IDF methods is convenient to discretize their real values. The MDL algorithm [11] based on information entropy minimization has been used.

4 Term Clustering

The term clustering is based on a special dictionary. The dictionary is defined as a total function

$$s : Terms \rightarrow Rep$$

which assigns just one representative from $Rep \subseteq Terms$ to each member of $Terms$ set. The s function defines equivalence classes on $Terms$ by equivalence relation σ:

$$(a, b) \in \sigma \iff s(a) = s(b)$$

Reversely, let $C \in Terms/\sigma$, there always exists some function s. If r is an arbitrary member of C, then

$$s(x) = r \quad \text{for all} \quad x \in C$$

The construction of dictionary consits of following steps:

1. Finding characteristic set for each term $t \in Terms$.
2. Defining equivalence classes on $Terms$ set based on similarity of their characteristic set.
3. Dictionary function s definition.

4.1 Characteristic set

Characteristic set construction is mostly based on using the Sketch Engine and its word sketches. Word sketches are one-page automatic, corpus-based summaries of a word's grammatical and collocational behavior generated by Sketch Engine which takes as input a corpus of any language and a corresponding grammar patterns and which generates word sketches for the words of that language [10].

It suggest itself to look for similar word sketches and build a thesaurus. For each lemma l with sufficient frequency we get a list of similar words $SP_l = [w_1, w_2, \ldots, w_n]$ ordered by their indexes of similarity $i_1, \ldots, i_n$ with lemma l [12]. Lets define the *characteristic list* $CHL(l)$ for each lemma l from the corpus:

- if frequency of lemma l in the corpus is less than 100:
  $CHL(l) = [l]$
- else:
  $CHS(l) = [w_1, w_2, \ldots, w_k] : \forall i_j \in \{i_1, i_2, \ldots i_k\} : i_j \geq 0.1$

An example of characteristic list of lemma *auto* (car) is shown in Table 1.

Table 1. Characteristic list of lemma *auto*

auto	1
automobil	0.184
autobus	0.171
vůz	0.166
vozidlo	0.153
vlak	0.141
aut	0.133
tramvaj	0.126
loď	0.124
letadlo	0.112
trolejbus	0.11

The table shows that the incorporated words are really semantically similar. However, there are some problems with homonyms and tagging errors (in this case term *aut*). The *characteristic set* is defined in the way of eliminating words occurred in the corpus more frequently in other senses than we currently treat with.

Let $CHL(l) = [w_1, w_2, \ldots, w_k]$ is the characteristic list of the lemma l, $S(l) = \{w_1, w_2 \ldots, w_k\}$ and $S_p(l) = \{w_i | i \leq k/p\}$ where $p \in \mathbb{R}^+$ is a constant coefficient. The *characteristic set* is defined as

$$CH(l) = \{w_i : q \cdot |S(w_i) \cap S_p(l)| \geq |S_p(l)|\}$$

where $q \in \mathbb{R}^+$ is an appropriate constant. The experiments have shown that the best values seem to be $p = 2, q = 2$.

4.2 Dictionary construction

When we have a characteristic set for each lemma from corpus it remains to define clustering and dictionary function s. Intuitively, the clusters are composed of terms with similar characteristic sets. In this work, the similarity is measured by Jaccard index, where similarity of terms a and b is defined as

$$j(a,b) = \frac{|CH(a) \cap CH(b)|}{|CH(a) \cup CH(b)|}$$

The clustering works on the principle of hierarchical clustering [13] using top-down method. Minimal similarity for joining sets was experimentally set to 0.45. These clusters define equivalence relation σ.

Let $freq(x)$ is a frequency of term x. We define dictionary function s: $\forall S \in Terms/\sigma, \forall a \in S : s(a) = b$ where $b \in S, freq(b) = max\{freq(x)|x \in S\}$. In the case of ambiguity the first possible lemma in lexicographical order is used.

Finally, when we have dictionary function s, we are able to replace all terms t in corpus by their representatives $s(t)$.

5 Attribute Selection

Even after application of the dictionary function there are a lot of different terms for using machine learning algorithms in the corpus and it is necessary to select the most convenient ones. Statistics provides some standard tools for testing if the class label and a single term are significantly correlated with each other. For simplicity, let us consider a binary representation of the model. Fix a term t and let

- $k_{i,0}$ = number of documents in class i not containing term t
- $k_{i,1}$ = number of documents in class $i$containing term t

This gives us a contingency matrix

$I_t \backslash$C	1	2	...	11
0	$k_{1,0}$	$k_{2,0}$	...	$k_{11,0}$
1	$k_{1,1}$	$k_{2,1}$	...	$k_{11,1}$

where C and I_t denote boolean random variable and $k_{l,m}$ denotes the number of observation where $C = l$ and $I_t = m$.

5.1 χ^2 test

This measure is a classical statistic approach. We would like to test if the random variables C and I_t are independent or not. The difference between observed and expected values is defined as:

$$\chi^2 = \sum_{l \in Class} \sum_{m \in \{0,1\}} \frac{(k_{l,m} - n \cdot P(C = l)P(I_t = m))^2}{n \cdot P(C = l)P(I_t = m)}$$

5.2 Mutual Information Score

This measure from information theory is especially useful when the multinominal document model is used and documents are of diverse length (as is usual). The mutual information score is defined as:

$$MI(I_t, C) = \sum_{l \in Class} \sum_{m \in \{0,1\}} \frac{k_{l,m}}{n} log \frac{k_{l,m}/n}{(k_{l,0} + k_{l,1}) \cdot (\sum_{i \in Class} k_{i,m})/n^2}$$

6 Classification and Evaluation

We have tested the classification using four algorithms (C4.5, *k*-nearest neighbors, Naîve Bayes classifier and Support machines) on 3,500 randomly chosen training samples and 1,500 testing examples. For testing has been used 10-fold cross validationi [14]. As an implementation, we have chosen open source data mining software Weka [15] for algorithm C4.5, *k*-nearest neighbors and Naîve Bayes classifier and LIBSVM [16] for Support Vector machines.

First, we compare preprocessing methods and selected machine learning algorithms on data without clustering and document extending. Next, the best-resulting method is chosen to test approaches presented in this paper. In Figure 1 you can see overall accuracy graphs of all presented algorithms and methods of document model representation. The best results with 79.04 % of overall accuracy have been acquired using Support vector machines algorithm, term frequency document model and MI-score selection of attributes.

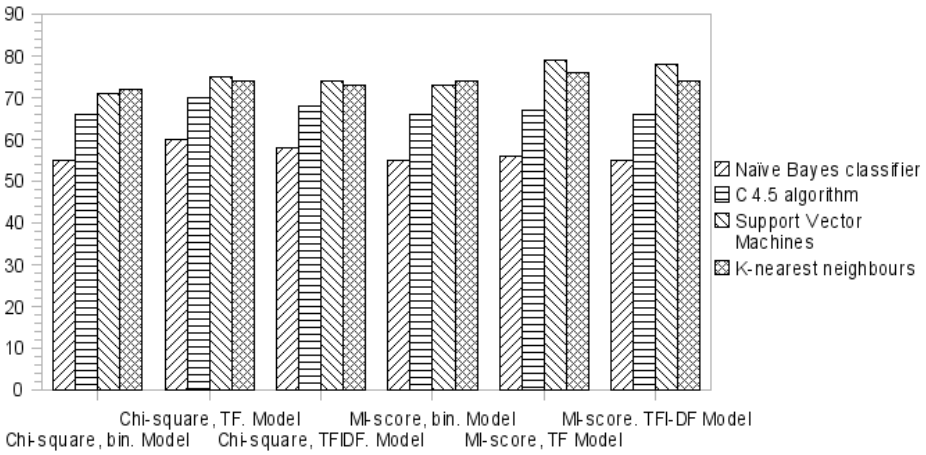


Fig. 1. Preprocessing and classification algorithms

Figure 2 shows dependency of overall accuracy on attribute number without clustering, with clustering based on same lemmas and with clustering

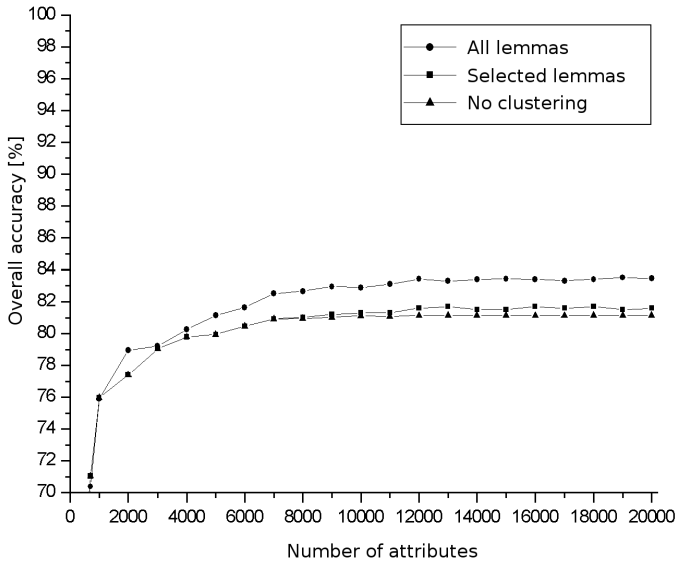


Fig. 2. Clustering methods

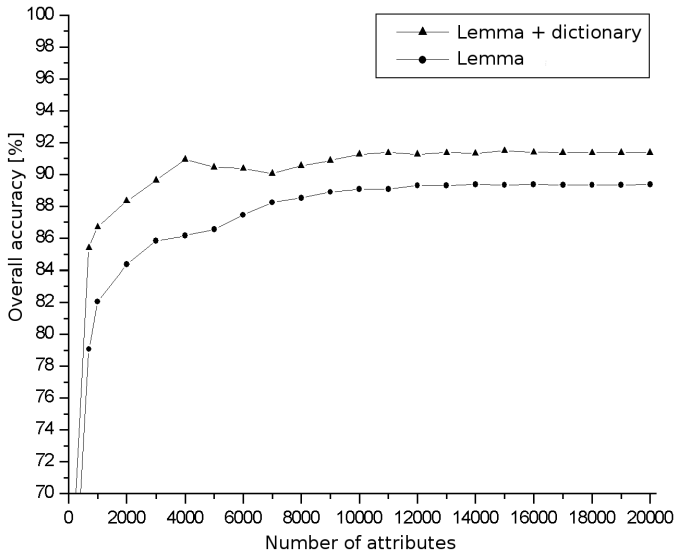


Fig. 3. Extending by referenced documents

based on selected lemmas. In the third case, only nouns, adjectives, verbs and adverbs have been selected. You can see that overall accuracy in all cases grows

till about 12,000 attributes. After this threshold the overall accuracy does not vary significantly. The best result (83.4 %) was acquired using clustering based on same lemmas.

Finally, Figure 3 shows result of experiments with extended documents, clustering based on same lemmas and on both lemmas and dictionary. The overall accuracy growth from previous experiment is about 5.9 % for lemma based clustering and 8.2 % for dictionary based clustering.

7 Conclusion

We have presented a method of automatic web page classification into given 11 semantic classes. Special attention has been laid on treating with short documents which often occur on the internet. There have been introduced two approaches which enable classification with overall accuracy about 91 %. Several machine learning algorithms and preprocessing methods have been tested. The best result has been acquired using Support vector machines with linear kernel function (followed by method of k-nearest neighbors) and term frequency document model with attribute selection by mutual information score.

Acknowledgments. This work has been partly supported by the Academy of Sciences of Czech Republic under the projects 1ET100300419 and 1ET200610406, by the Ministry of Education of CR within the Center of basic research LC536 and in the National Research Programme II project 2C06009 and by the Czech Science Foundation under the project 407/07/0679.

References

1. Asirvatham, A.P., Ravi, K.K.: Web page categorization based on document structure (2008) `http://citeseer.ist.psu.edu/710946.html`.
2. Santini, M.: Some issues in automatic genre classification of web pages. In: JADT 2006 – 8èmes Journèes internationales d'analyse statistiques des donnés textuelles, University of Brighton (2006).
3. Mladenic, D.: Turning Yahoo to automatic web-page classifier. In: European Conference on Artificial Intelligence. (1998) 473–474.
4. Pierre, J.M.: On automated classification of web sites. **6** (2001) `http://www.ep.liu.se/ea/cis/2001/000/`.
5. Tsukada, M., Washio, T., Motoda, H.: Automatic web-page classification by using machine learning methods. In: Web intelligence: research and development, Maebashi City, JAPON (23/10/2001) (2001).
6. Li, S., Momoi, K.: A composite approach to language/encoding detection. 9th International Unicode Conference (San Jose, California, 2001).
7. Sedláček, R.: Morphemic Analyser for Czech. Ph.D. thesis, Faculty of Informatics, Masaryk University, Brno (2005).
8. Šmerk, P.: Towards Morphological Disambiguation of Czech. Ph.D. thesis proposal, Faculty of Informatics, Masaryk University, Brno (2007).

9. Rychlý, P.: Korpusové manažery a jejich efektivní implementace (in Czech). Ph.D. thesis, Faculty of Informatics, Masaryk University, Brno (2000).
10. Kilgarriff, A., Rychlý, P., Smrž, P., Tugwell, D.: The Sketch engine in practical lexicography: A reader. (2008) 297–306.
11. Fayyad, U.M., Irani, K.B.: On the handling of continuous-valued attributes in decision tree generation. Machine Learning **8** (1992) 87–102.
12. Kilgarriff, A.: Thesauruses for natural language processing. Proc NLP-KE (2003).
13. Berka, P.: Dobývání znalostí z databází. Academia (2003).
14. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: IJCAI. (1995) 1137–1145.
15. Witten, I.H., Frank, E.: Data mining: Practical machine learning tools and techniques. Technical report, Morgan Kaufmann, San Francisco (2005).
16. Chang, C.C., Lin, C.J.: LIBSVM: a Library for Support Vector Machinesi. Technical report, Department of Computer Science National Taiwan University, Taipei 106, Taiwan (2007).

Building Big Czech Corpus
Collecting and Converting Czech Corpora

Pavel Hančar

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

Abstract. This paper describes creating of a big Czech corpus from many Czech corpora kept on the NLP Centre server. It describes new tools developed for this purpose, difficulties which may come up and a way how solve them.

1 Introduction

A corpus is a large collection of texts in a certain language. NLP Centre has many Czech corpora, but doesn't have any big one. A very valuable aspect of corpora is a morphological tagging, but this is missing in current data. So the task is to collect all Czech corpora (let's call them input corpora), to do the tagging, and to compile the result for corpus manager.

Corpora contains two kinds of data: the text itself in a vertical form and information about the text (metadata). The vertical form means, that every word or punctuation mark has it's own line (there are some special cases e.g. "..." is one position, not three). Documents, paragraphs, etc. are determined by XML marks [1,2].

Generally, the data are kept as text files, but there is more possibilities how to do this. Most of the input corpora consist of many vertical files, where each of them has it's metadata file. However the format we need is whole corpus in one vertical file with metadata in heading of each document. It is format used by the compiler ENCODEVERT. This program compiles corpora to binary files usable for the corpus manager MANATEE.

2 Corpus conversion

Conversion means concatenation of vertical files (one can contain more documents), looking for metadata of each document and creating document heading tag. This task has these specifics:

Language: Some corpora contain a few documents in different language, than the most common is. In that case the information about the document language is mentioned in metadata, and along this data field the documents are filtered.

Petr Sojka, Aleš Horák (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2008, pp. 94–97, 2008.

Multivalues in metadata: Some fields in metadata files can have more values. The specification [3] allows only the one-line notation:

```
M_Date: 2000-05-07; 2000-06-22
```

But in real data is possible to see this:

```
M_Date: 2000-05-07
M_Date: 2000-06-22
```

Also in XML it's not possible to use more attributes with the same name. So the right way for ENCODEVERT is `m_date="2000-05-07;2000-06-22"`. Naturally we must be careful in the case of language attribute if we need only one language.

Syntactic errors: There are syntactical errors in verticals and in metadata. For example this can appear in vertical as one position:

```
...stalo
```

Another problem are XML tags divided character after character to more lines, or bad order of XML tags eventually their bad occurrence. In metadata e.g. one data field on more lines can appear.

2.1 Implementation

The implementation on the NLP Centre's server is a set of Python and Shell scripts available in `/corpora/priprava_dat/ib047_99-07/scripts`.

vertjoin.py `Usage: vertjoin.py [-l lang] DIRECTORY [OUTPUT_FILE]`
Main script walking through directory tree with the root `DIRECTORY`, looking for `*.vert` and corresponding `*.meta` files and concatenating them on the standard output or to the `OUTPUT_FILE`.
Script expects verticals with document tags named as `doc` with obligatory attribute `id` corresponding to field `Doc` in metadata.
`vertjoin.py` also implements two methods to repair easy syntactic errors:

normal_meta A method for metadata which removes a possible backslash on the and of line and joins a data field written on two lines.

normal_vert A method for verticals which removes empty lines, strips possible white-spaces around the position, divides more words on one line, ensures the "..." not to be in the same position with some word and puts together short broken tags (means the tags without attributes e.g. </p>).

predesamb.sh `Usage: predesamb.sh INPUT_FILE`
Script repairing main syntactic errors of verticals. It's pipeline of smaller scripts. These scripts repair errors concerning concrete tags. It looks like this:

```
cat $1 |p_doc.py |p_p.py|p_tag.py|note_p.py|\
more_closing_tags.py|gg.py|g_tag_g.py|sed 's/</*q>/"/g'
```

Last `sed` command replaces <q> and </q> by symbol ". The q tag is specified in [2], but it's not accepted by ENCODEVERT.

3 Corpora tagging

A very important aspect of corpora is the morphological tagging. It makes corpora being especial tools even in Google ages. NLP Centre has it's own tagger for Czech language named DESAMB. It's developed by Pavel Šmerk and based on partial syntactic analyser DIS/VADIS developed by Eva Mráková-Žáčková.

DIS/VADIS is written in Prolog, which is a disadvantage, because it's quite slow. Rest of the DESAMB is written in Perl, but DIS/VADIS slows down whole process of tagging. So a future Pavel Šmerk's plan is to rewrite DIS/VADIS also to Perl.

Next disadvantage of Prolog is probably a faulty allocation of memory. It seems, that DESAMB is not able to process big corpora, because then the Prolog fails due to lack of memory. This problem appeared on verticals about 20 million positions long, but in this case, the second aspect was quite complicated structure of vertical (many tags, tables, long sentences, etc.).

So it's needed to divide verticals into more smaller parts before processing. A script `divide.py` implements this function, but it probably won't be included to DESAMB, because future Perl implementation of DESAMB doesn't need that.

divide.py `Usage: divide.py [-l lines] [-t doc_tag] FILE1 [FILE2 ...]`

Where `lines` is count of lines which is possible to shorten by one of letters `KkMm` (eg. `30K` means thirty thousands, `2m` means two million). The default value is 1 million. Default value of `doc_tag` is `"doc"`.

This script divides verticals – after counting `count_of_lines` – on the nearest document border. Output files are written in current directory having original name extended by three-figure number of part (`FILE1.001`, `FILE1.002`, ...).

The script can have more input files and also it can read from standard input.

Nevertheless, the tagging of corpora after dividing them is also slow. It becomes evident on the 20 million positions long verticals . It was tested on five such corpora and usually one of the corpora took about one day on one computer. But there was the especial one, processed about five days. Probable cause of this is count of long sentences (including also enumerations or tables without punctuation marks). All the corpora have a few sentences over 500 words long, but the 5-days one has about 460 of these sentences.

Last but not least current DESAMB has problems with parsing of corpora e.g. considering XML tags to be a word. The question is, if it is only because of complicated source code of parser, or if it can't be better because of too expressive syntax of corpora with vague definition.

A meaningful goal seems to be an improving tagger so, that its output would be usable for ENCODEVERT. But nowadays DESAMB would cause many warnings in ENCODEVERT, which can be prevented by a script `postdesam.sh`.

postdesam.sh A script repairing syntactic errors on the output of DESAMB. It also removes <s> and </s>, that are tags added in DESAMB to determinate sentence borders. Main part of the script is a pipeline consisting of `sed` substitute commands:

```
cat "$in"| sed '/^<\/*s> *$/d'| sed 's/<\/*s>//g' |\
 sed 's/^$/<l>/g' | sed 's/\(^<doc.*>\)\s*<doc.*$/\1/g'|\
 sed 's/<\(\/*[^<>][^<>]*\)>[\t ]*<\/*[^<>][^<>]*>[\t ]k?/<\1>/g'\
 > "$out"
```

Maybe it prevents more bugs than needed, because some substitutes were added during of changes in DESAMB.

4 Conclusion

This paper describes, some problems with building big corpora and shows a way how to solve them. Described way has its first result. It is a 80 million positions Czech corpus consisting of data collected by students in Pavel Rychly's course IB047 Introduction to Corpus Linguistics and Computer Lexicography.

Future plans are clear – to collect next data to the corpus. Hopefully it will be easier than the first 80 million, because now the tools are ready and other corpora probably contain more consistent data than corpora created by many students.

Acknowledgments. This work has been partly supported by the Academy of Sciences of Czech Republic under the project 1ET200610406.

References

1. Jak vytvořit korpus (2000) `http://nlp.fi.muni.cz/cs/Jak_vytvorit_korpus`.
2. Popis vertikálu (2000) `http://nlp.fi.muni.cz/cs/Popis_vertikalu`.
3. Popis metainformací (2000) `http://nlp.fi.muni.cz/cs/Popis_metainformaci`.

Towards Natural Natural Language Processing
A Late Night Brainstorming

Petr Sojka

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

Abstract. An essay about mimicking some aspects of language processing in our heads, using information fusion and competing patterns.

1 Introduction

Usual approach to Natural Language Processing separates language processing into word form, morphological, syntactic, semantic and pragmatic levels. Most often processing of these levels are independent, and result of one level is communicated to the other unnaturally disambiguated to cut off less probable (but often linguistically valid) intermediate results (e.g. sentence syntactical parse trees) just to simplify things. Even though ungrammatical sentences are often used for communication between people (English as the second language), they are banned by NLP software. Considerable effort is given to the balancing general purpose corpora to choosen only such text examples, aiming at handling only [syntactically] correct language parts. Given that, for purposes of handling non-polished texts, blogs or even speech, these data resources fail badly, as the tools are trained and fine-tuned to the different type of input than used when processing real [speech] data.

As simple as possible, but not simpler. – Albert Einstein

2 Levels of Processing, Level Interaction and Importance of Complexity

Most of today's available texts is processed on *word form* level only (Google), with PageRank optimizing access to the most credible ones. Texts sharing the same forms are collected together and only the most credible picked up and shown. This suits most, but not all purposes.

Good *morphological* tools allows handling of *all* possible morphological categories, allowing their pruning in further stages of processing (syntactic analysis, etc.). The disambiguation should not be driven by pure statistics in applications like guesser.

Syntactic analysis aiming at only one (best) parsing tree, independently of sentence context, document type and other information is simply wrong.

Petr Sojka, Aleš Horák (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2008, pp. 98–100, 2008.

Analysis of potentially billions of possible trees of long sentences is waste of computer resources. Most probable partial parse trees are collected together and shown as the parsed result for further processing. Much better approach is to collect possible sentence segmentations of main building blocks (phrases) and not limiting the analysis outcome to the correct full sentence parses only.

Another bottleneck of today NLP processing is *semantics* handling. Bubble of semantic net starts to blow out, as there is not single semantic representation suitable for all purposes and applications. Linguistic resources are scarce, and wordnets lack many important aspects as deduction and thematic folding (specific domain adaptation and usage, with exception of framenet). Promising formalisms like TIL need necessary language resources.

Little attention is given to *pragmatics* in NLP, as a starter of disambiguation process. Disambiguation, at all levels, should be driven by the final application, deriving from the purpose, classification type of communicated text, intertwisting and backtracking between all levels of linguistic processing. The tools should not be trivialized and should handle multiple lemmata, parses, meanings. Language handling may be as complex as the life it describes, not simpler.

Be as elegant as the situation will allow.

3 Information Fusion and Patterns

The suggested remedy to the current status quo is the design of a modular NLP system for parallel language processing at different levels, allowing mutual interactions and processing between data structures and intermediate results at all levels. The data structures may be not only grammar chunks, framenets and wordnets, but also empirical evidence of language usage (text corpora processed), allowing pattern matching of linguistic data and knowledge representation at various, but interlinked levels.

For several purposes in this scenario, *competing patterns* [1,2] may be used: sentence or phrase segmentation (alphabet is word forms or lemmas), morphological disambiguation patterns (alphabet is gramatical categories and lemmata) [3], and even pragmatics patterns (alphabet being events in time and meaning terms). Same terms in pattern alphabets used will allow for connecting information on different level of language processing – the patterns may be derived from available text and dialogue corporas [4,5]. Pattern storage in the packed digital trie is very compact and allow blindingly fast language data retrieval at the constant time (limited by the pattern length only, e.g. by width of [local] context covered).

4 Conclusion

In this paper, we have presented several thoughts about current state of the art of natural language processing approaches, and have outlined several

directions of improvement towards 'natural' way of text processing, grabbing some metaphors from what is known about language processing in our brains.

Acknowledgments. This work has been partially supported by the Academy of Sciences of Czech Republic under the projects 1ET208050401, 1ET200190513 and by the Ministry of Education of CR within the Centre of basic research LC536 and National Research Programme 2C06009.

References

1. 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), pp. 157–162.
2. Sojka, P.: Competing Patterns in Language Engineering and Computer Typesetting. Ph.D. thesis, Masaryk University, Brno (2005).
3. Macháček, D.: Přebíjející vzory ve zpracování přirozeného jazyka (Competing Patterns in Natural Language Processing). Master's thesis, Masaryk University, Brno, Faculty of Informatics, Brno, Czech Republic (2003).
4. Antoš, D., Sojka, P.: Pattern Generation Revisited. In: Pepping, S., (Ed.): Proceedings of the 16th European TEX Conference, Kerkrade, 2001, Kerkrade, The Netherlands, NTG (2001) pp. 7–17.
5. 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) pp. 65–72.

Author Index

Book orders should be addressed to:

Pavel Mareček c/o FI MU
Botanická 68a
CZ-602 00 Brno
Phone: ++420 549 498 735
Email: marecek@kup.to

RASLAN 2008

Recent Advances in Slavonic Natural Language Processing
Second Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2008
Karlova Studánka, Czech Republic, December 5–7, 2008
Proceedings

P. Sojka, A. Horák (Eds.)

Published by Masaryk University, Brno in 2008

Cover design: Petr Sojka

Typesetting: Petr Sojka

Data conversion: Adam Rambousek

Printing: http://librix.eu

First edition, 2008

Serial number INF-4/08-02/58

ISBN 978-80-210-4741-9