RASLAN 2007
Recent Advances in Slavonic Natural Language Processing

Masaryk University
`http://nlp.fi.muni.cz/raslan/2007/`

P. Sojka, A. Horák (Eds.)

RASLAN 2007

Recent Advances in Slavonic Natural Language Processing

First Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2007
Karlova Studánka, Czech Republic,
December 14–16, 2007
Proceedings

Masaryk University
Brno 2007

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 2007 (1. : Karlova Studánka, Česko)

RASLAN 2007 : Recent Advances in Slavonic Natural Language Processing :
first workshop on ..., Karlova Studánka, Czech Republic,
December 14-16, 2007 : proceeedings / P. Sojka, A. Horák (eds.).
- 1st ed. - Brno : Masaryk University, 2007. - viii + 118 s.

ISBN 978-80-210-4471-5

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]
```

ISBN 978-80-210-4471-5

Preface

This volume contains the Proceedings of the First Workshop on Recent Advances in Slavonic Natural Language Processing (RASLAN 2007), organized by the the Center of Natural Language Processing at the Faculty of Informatics, Masaryk University and held on December 14th–16th 2007 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 Center. 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 2007 offers a rich program of presentations, short talks, technical papers and mainly discussions. A total of 14 papers were accepted, contributed altogether by 17 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. In addition we would like to thank Dagmar Janoušková who took care of the administrative burden with great efficiency, and contributed substantially to the detailed preparation of the conference. 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 2007 Karel Pala

Table of Contents

Part I

Morphological and Syntactic Parsing

DEB Platform Deployment
Current Applications

Aleš Horák and Adam Rambousek

Faculty of Informatics, Masaryk University Brno
Botanická 68a, 602 00 Brno, Czech Republic
hales@fi.muni.cz, xrambous@fi.muni.cz
http://deb.fi.muni.cz/

Abstract. In this paper, we summarize the latest development regarding the client dictionary writing applications based on the DEB II development platform. The DEB II framework is nowadays used in several full grown projects for preparation of high quality lexicographic data created within (possibly distant) teams of researchers.
We briefly present the current list of DEB II applications with the relevant projects and their phases. For each of the applications, we offer display the view of the interface with overview description of the most important features.

Key words: DEB platform, dictionary editor and browser, dictionary writing systems

1 Introduction

The Dictionary Editor and Browser (DEB) was first designed as a standalone program for writing dictionaries. After several problems with adaptation of the tool for coming new requirements, the second version, sometimes referred to as DEB II became a complete rewrite of the system based on open standards.

In the following text, we enlist the current state of DEB II applications. We believe that this list is the best evidence of the qualities of the framework as a whole together with several hundreds of DEB II users all over the world.

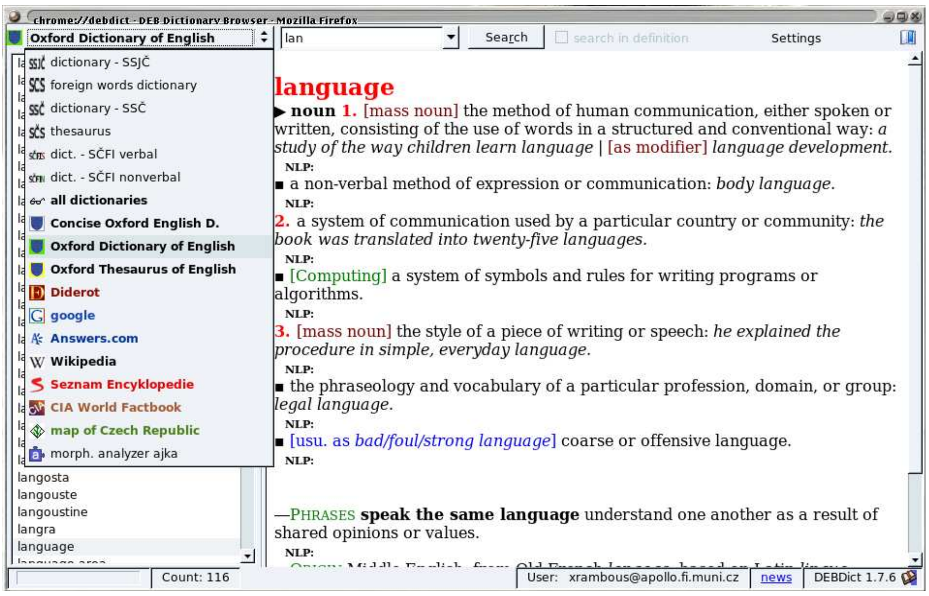
2 Current List of Implemented DEB Applications

In the following sections we present summary details of the particular DEB clients that are currently being implemented within the DEB platform.

2.1 DEBDict – General Dictionary Browser

This DEB client demonstrates several basic functions of the system:

– multilingual user interface (English, Czech, others can be easily added)



language

▶ **noun** 1. [mass noun] the method of human communication, either spoken or written, consisting of the use of words in a structured and conventional way: *a study of the way children learn language* | [as modifier] *language development.*

NLP:

■ a non-verbal method of expression or communication: *body language.*

NLP:

2. a system of communication used by a particular country or community: *the book was translated into twenty-five languages.*

NLP:

■ [Computing] a system of symbols and rules for writing programs or algorithms.

NLP:

3. [mass noun] the style of a piece of writing or speech: *he explained the procedure in simple, everyday language.*

NLP:

■ the phraseology and vocabulary of a particular profession, domain, or group: *legal language.*

NLP:

■ [usu. as *bad/foul/strong language*] coarse or offensive language.

NLP:

—PHRASES **speak the same language** understand one another as a result of shared opinions or values.

NLP:

Fig. 1. The DEBDict common interface to several dictionaries with different structures.

– queries to several XML dictionaries (of different underlying structure) with the result passed through an XSLT transformation
– connection to Czech morphological analyzer
– connection to an external website (Google, Answers.com)
– connection to a geographical information system (display of geographical links directly on their positions within a cartographic map) or any similar application

The version of DEBDict that is currently running on our server provides a common interface to 7 dictionaries (see the Figure 1):

– the Dictionary of Literary Czech Language (SSJČ, 180,000 entries)
– the Dictionary of Literary Czech (SSČ, 49,000 entries)
– the Reference Dictionary of Czech Language (PSJČ, 200,000 entries)
– the Dictionary of foreign words (46,000 entries)
– the Dictionary of Czech Synonyms (thesaurus, 23,000 entries)
– two dictionaries of Czech Phrasal Words and Idioms (4,000 and 10,000 entries)
– the Diderot encyclopedia (90,000 entries)

As an addition, DEBDict features an interconnection to several web systems and the geographical system with the list of the Czech towns and cities.

DEBDict is also able to solve common problems with publication copyright for particular dictionaries – DEBDict supports individual user access rights.

This means that it is possible to display selected dictionaries only to a limited group of users.

2.2 DEBVisDic

DEBVisDic was one of the first applications built over the DEB II platform – it was designed as a completely new client-server tool for WordNet browsing and editing.

DEBVisDic uses new versatile interface (see the Figure 2) that allows the user to arrange the work without any limitations. Of course, DEBVisDic contains all the main features that were present in VisDic:

– multiple views of multiple WordNets
– freely defined text views
– synset editing
– hypero-hyponymic tree
– query result lists
– plain XML view of a synset
– synchronization
– inter-dictionary linking
– tree browsing
– consistency checks
– journaling

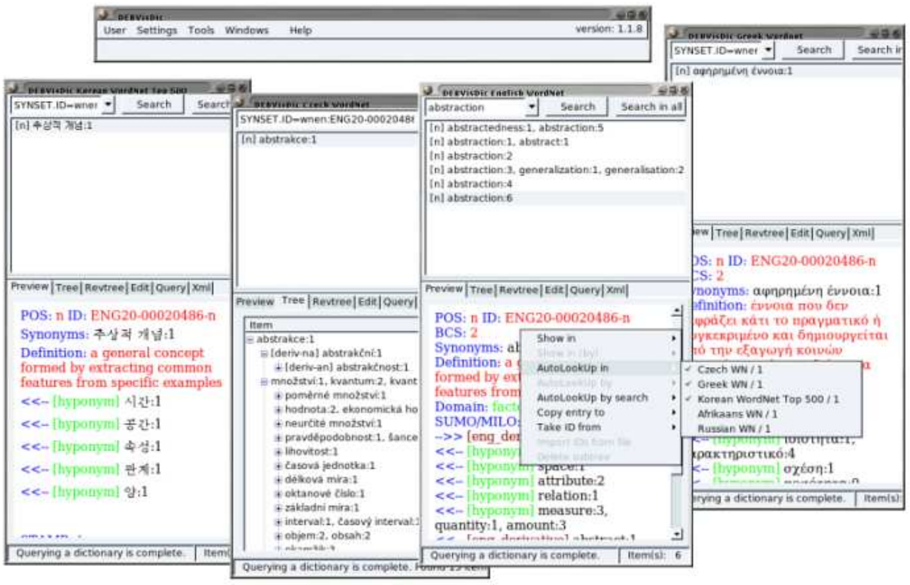
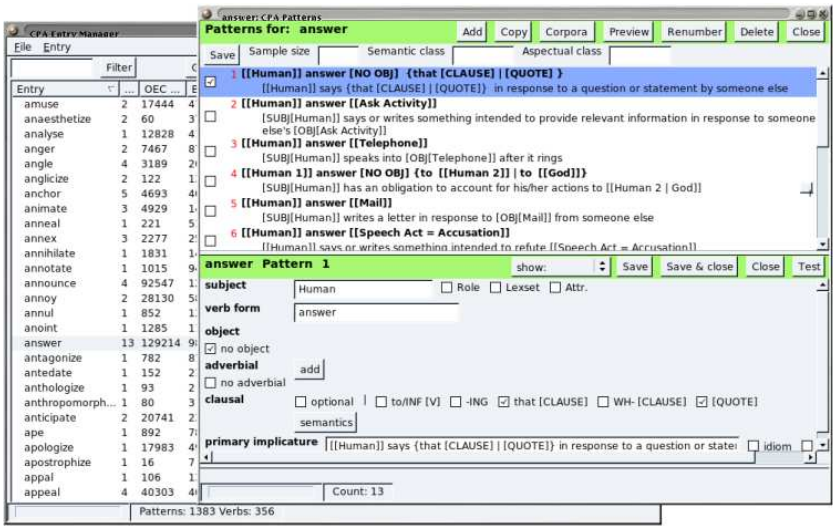


Fig. 2. The DEBVisDic main interface



CPA Entry Manager

File Entry

Filter

Entry	...	OEC ...	E
amuse	2	17444	4
anaesthetize	2	60	3
analyse	1	12828	4
anger	2	7467	8
angle	4	3189	2
anglicize	2	122	1
anchor	5	4693	4
animate	3	4929	1
anneal	1	221	5
annex	3	2277	2
annihilate	1	1831	1
annotate	1	1015	9
announce	4	92547	1
annoy	2	28130	5
annul	1	852	1
anoint	1	1285	1
answer	13	129214	9
antagonize	1	782	8
antedate	1	152	2
anthologize	1	93	2
anthropomorph...	1	80	3
anticipate	2	20741	2
ape	1	892	7
apologize	1	17983	4
apostrophize	1	16	7
appal	1	106	1
appeal	4	40303	4

Patterns: 1383 Verbs: 356

answer: CPA Patterns

Patterns for: answer

Sample size · Semantic class · Aspectual class

1 **[[Human]] answer [NO OBJ] {that [CLAUSE] | [QUOTE] }**
[[Human]] says {that [CLAUSE] | [QUOTE]} in response to a question or statement by someone else

2 **[[Human]] answer [[Ask Activity]]**
[SUBJ[Human]] says or writes something intended to provide relevant information in response to someone else's [OBJ[Ask Activity]]

3 **[[Human]] answer [[Telephone]]**
[SUBJ[Human]] speaks into [OBJ[Telephone]] after it rings

4 **[[Human 1]] answer [NO OBJ] {to [[Human 2]] | to [[God]]}**
[SUBJ[Human]] has an obligation to account for his/her actions to [[Human 2 | God]]

5 **[[Human]] answer [[Mail]]**
[SUBJ[Human]] writes a letter in response to [OBJ[Mail]] from someone else

6 **[[Human]] answer [[Speech Act = Accusation]]**
[[Human]] says or writes something intended to refute [[Speech Act = Accusation]]

answer Pattern 1

show:

subject Human ☐ Role ☐ Lexset ☐ Attr.

verb form answer

object
☑ no object

adverbial
☐ no adverbial

clausal ☐ optional ☐ to/INF [V] ☐ -ING ☑ that [CLAUSE] ☐ WH- [CLAUSE] ☑ [QUOTE]

primary implicature [[Human]] says {that [CLAUSE] | [QUOTE]} in response to a question or state ☐ idiom ☐

Count: 13

Fig. 3. The DEB CPA tool.

– user configuration

With the help of the DEB platform reusability, DEBVisDic will be supplemented with many new features that are currently accessible only as separate tools or resources. This functionality includes:

– connection to a morphological analyzer (for languages, where it is available)
– connection to language corpora, including Word Sketches statistics
– access to any electronic dictionaries stored within the DEB server
– searching for literals within encyclopedic web sites
– and many others

Currently, DEBVisDic is also used for preparation of new Polish, Hungarian, Slovenian, Dutch (within the Cornetto project), Nepalese and Afrikaans WordNets and it is proposed as the main tool for the prepared Global WordNet Grid.

2.3 DEB CPA Editor and Browser

Corpus Pattern Analysis (CPA, [1]) is a new technique for mapping meaning to words in text. No attempt is made in CPA to identify the meaning of a verb or noun directly, as a word in isolation. Instead, meanings are associated with prototypical sentence contexts. Concordance lines are grouped into semantically motivated syntagmatic patterns. Associating a "meaning" with each pattern is a secondary step, carried out in close coordination with the assignment of concordance lines to patterns.

CPA editing tool (see the Figure 3) displays the list of verb entries, along with the information who and when updated the entry. Each entry consists of several patterns (the number of patterns is not limited) and it is possible to freely modify their order and content. The main part of the tool, the pattern editing window, allows to enter and modify all the information about one pattern. The form is very versatile, e.g. it allows to add any number of subject/object alternations. The tool is connected to an on-line resource – it is possible to look up subject and object semantic type in Brandeis Semantic Ontology [2] which is hosted on a web server at Brandeis University. Examples documenting the pattern are taken from BNC using a modified version of Bonito2 corpus manager that is integrated to the DEB CPA tool.

2.4 DEB TEDI Terminological Dictionary Tool

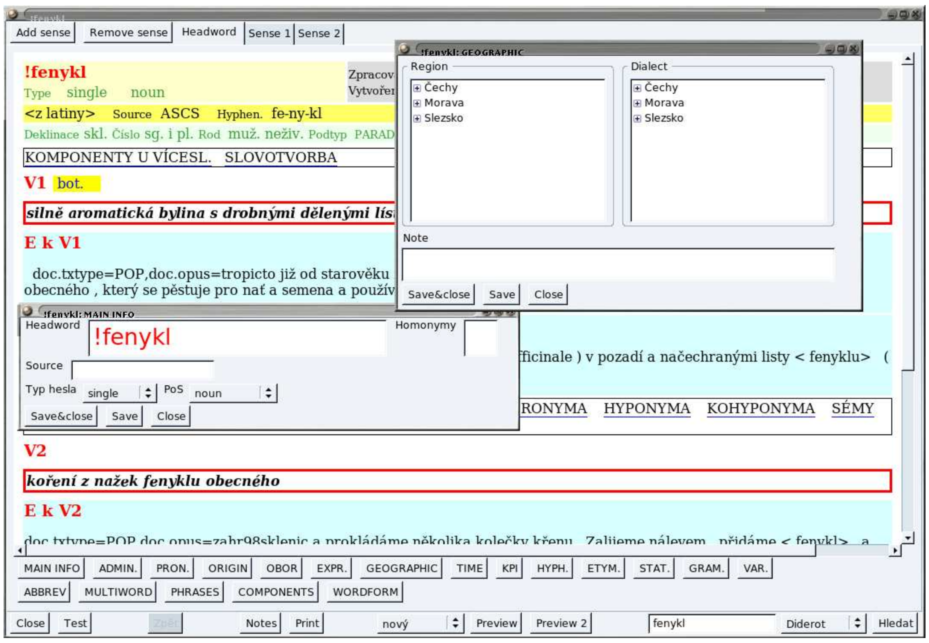
The DEB TEDI client is the main tool used for preparation of a new terminological dictionary of Czech art terms. This work is a joint project of the Faculty of Fine Arts, Brno University of Technology and Masaryk University. The aim of the project is to build a terminological database consisting of about 5 000 dictionary entries which are classified into categories and supplemented with term definitions, translations info English, German and French, and with Czech usage examples. The resulting dictionary will be offered as a publicly available application directed especially to fine arts students.

2.5 The PRALED Lexicographic Station

This client is designed for the development of the Czech Lexical Database (CLD, denoted also as LEXIKON 21 [3]) and it serves as a main tool in preparation of the new comprehensive and exhaustive database of lexicographic information for the Czech language. The user's part of the PRALED tool is presently under the development in the Institute of Czech Language (ICL), Czech Academy of Sciences, Prague.

The PRALED system offers the following functionality:

– queries to several XML dictionaries (of different underlying structures), particularly to all relevant Czech dictionaries, i.e. SSJČ, SSČ, PSJČ, SCS, SČFI and DIDEROT (see [4,5,6,7,8]),
– editing existing or writing new dictionary entries. A lexicographer can use a set of forms which define the structure of the entry and fill in all relevant fields (see the Figure 4) which presently are:
  • orthoepy (spelling)
  • morphological properties (POS, the respective grammatical categories)
  • description of the meaning (entry definition)
  • word formation nest (subnet)
  • syntactic properties (most often valencies)
  • stylistic, domain and regional features
  • semantic relations to other entries (cross-references)



!fenykl
Type single noun
<z latiny> Source ASCS Hyphen. fe-ny-kl
Deklinace skl. Číslo sg. i pl. Rod muž. neživ. Podtyp PARAD
Zpracov
Vytvořen
KOMPONENTY U VÍCESL. SLOVOTVORBA
V1 bot.
silně aromatická bylina s drobnými dělenými lís
E k V1
doc.txtype=POP,doc.opus=tropicto již od starověku
obecného , který se pěstuje pro nať a semena a použív
fficinale) v pozadí a načechranými listy < fenyklu> (
RONYMA HYPONYMA KOHYPONYMA SÉMY
V2
koření z nažek fenyklu obecného
E k V2
doc.txtype=POP,doc.opus=zahr98sklenic a prokládáme několika kolečky křenu . Zalijeme nálevem , přidáme < fenykl> a

Region: Čechy, Morava, Slezsko
Dialect: Čechy, Morava, Slezsko

Headword: !fenykl
Typ hesla: single
PoS: noun

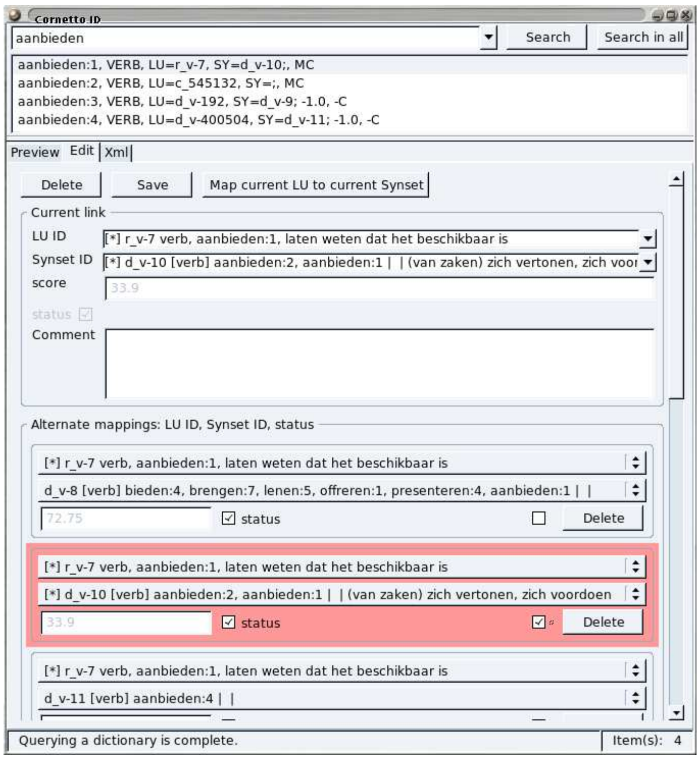
nový
fenykl
Diderot

Fig. 4. The PRALED user interface

- etymological information
- integration with Czech morphological analyzer
- connection to an external website (Google, Answers.com)
- remarks and additional comments
- integration with the corpus manager Bonito2 and Word Sketch Engine [9], which allows a lexicographer to obtain the sorted individual word contexts including frequencies and statistical distribution parameters (salience).

2.6 Cornetto

The Cornetto project (STE05039) is funded by the Nederlandse Taalunie in the STEVIN framework. The goal is to build a lexical semantic database for Dutch, covering 40K entries, including the most generic and central part of the language. Cornetto will combine the structures of both the Princeton WordNet and FrameNet for English [10], by combining and aligning two existing semantic resources for Dutch: the Dutch WordNet [11] and the Referentie Bestand Nederlands [12]. The Dutch WordNet (DWN) is similar to the Princeton WordNet for English, and the Referentie Bestand (RBN) includes frame-like information as in FrameNet plus additional information on the combinatoric behaviour of words in a particular meaning. The combination of the two lexical resources will result



aanbieden:1, VERB, LU=r_v-7, SY=d_v-10;, MC
aanbieden:2, VERB, LU=c_545132, SY=;, MC
aanbieden:3, VERB, LU=d_v-192, SY=d_v-9; -1.0, -C
aanbieden:4, VERB, LU=d_v-400504, SY=d_v-11; -1.0, -C

Current link

LU ID	[*] r_v-7 verb, aanbieden:1, laten weten dat het beschikbaar is
Synset ID	[*] d_v-10 [verb] aanbieden:2, aanbieden:1 | | (van zaken) zich vertonen, zich voor
score	33.9
status	✓
Comment	

Alternate mappings: LU ID, Synset ID, status

[*] r_v-7 verb, aanbieden:1, laten weten dat het beschikbaar is
d_v-8 [verb] bieden:4, brengen:7, lenen:5, offreren:1, presenteren:4, aanbieden:1 | |
72.75 ✓ status ☐ Delete

[*] r_v-7 verb, aanbieden:1, laten weten dat het beschikbaar is
[*] d_v-10 [verb] aanbieden:2, aanbieden:1 | | (van zaken) zich vertonen, zich voordoen
33.9 ✓ status ✓ Delete

[*] r_v-7 verb, aanbieden:1, laten weten dat het beschikbaar is
d_v-11 [verb] aanbieden:4 | |

Querying a dictionary is complete. Item(s): 4

Fig. 5. Cornetto Identifiers window, showing the edit form with several alternate mappings

in a much richer relational database that may improve natural language processing (NLP) technologies, such as word sense-disambiguation and language-generation systems. In addition to merging the WordNet and FrameNet-like information, the database is also mapped to a formal ontology to provide a more solid semantic backbone.

The resulting data structure is stored in a database that keeps separate collections for lexical units (mainly derived from RBN), synsets (derived from DWN) and a formal ontology (SUMO/MILO plus extensions [13]). These 3 semantic resources represent different view points and layers of linguistic, conceptual information. The alignment of the view points is stored in a separate mapping table. The database is itself set up so that the formal semantic definition of meaning can be tightened for lexical units and synsets by exploiting the semantic framework of the ontology. At the same time, we want to maintain the flexibility to have a wide coverage for a complete lexicon and encode additional linguistic information. The resulting resource will be made available in the form of an XML database.

The Cornetto database (CDB) consists of 3 main data collections:

1. Collection of Lexical Units, mainly derived from the RBN
2. Collection of Synsets, mainly derived from DWN
3. Collection of Terms and axioms, mainly derived from SUMO and MILO

In addition to the three data collections, a separate table of so-called Cornetto Identifiers (CIDs) is provided, see the Figure 5. These identifiers contain the relations between the lexical units and the synsets in the CDB but also to the original word senses and synsets in the RBN and DWN.

Since one of the basic parts of the Cornetto database is the Dutch WordNet, we have decided to use DEBVisDic as the core for Cornetto client software. We have developed four new modules, described in more details below. All the databases are linked together and also to external resources (Princeton English WordNet and SUMO ontology), thus every possible user action had to be very carefully analyzed and described.

3 Conclusions

During the last three years, DEB II has been going through rapid development and several real applications for electronic dictionaries have been built. The free access to the Brno DEB server is nowadays in use by more than 250 users from 14 countries.

The DEB II server part is also available for download and is currently installed on 7 servers worldwide (Brno, Prague, Amsterdam-UvA, Amsterdam-VU, Poznan, Johannesburg and Budapest), where the DEB applications are used for national research projects.

Acknowledgments

This work has been partly supported by the Academy of Sciences of Czech Republic under the project T100300419 and by the Ministry of Education of CR within the Center of basic research LC536.

References

1. Hanks, P.: Corpus Pattern Analysis. In: Proceedings of the Eleventh EURALEX International Congress, Lorient, France, Universite de Bretagne-Sud (2004).
2. Pustejovsky, J., Havasi, C., Littman, J., Rumshisky, A., Verhagen, M.: Towards a Generative Lexical Resource: The Brandeis Semantic Ontology. In: Proceedings of LREC 2006, Genoa, Italy (2006) demo.
3. Rangelova, A., Králík, J.: Wider Framework of the Research Plan Creation of a Lexical Database of the Czech Language of the Beginning of the 21st Century. In: Proceedings of the Computer Treatment of Slavic and East European Languages 2007, Bratislava, Slovakia (2007) 209–217.

4. Petr, J., et al.: Slovník spisovného jazyka českého (Dictionary of Written Czech, SSJČ). 1st edn. Academia, Praha (2002) electronic version, created in the Institute of Czech Language, Czech Academy of Sciences Prague in cooperation with Faculty of Informatics, Masaryk University Brno.
5. Filipec, J., et al.: Slovník spisovné češtiny (Dictionary of Literary Czech, SSČ). 1st edn. Academia, Praha (1995) electronic version, LEDA, Praha.
6. Havránek, B., ed.: Příruční slovník jazyka českého (Reference Dictionary of Czech Language, PSJČ). Státní nakladatelství/SPN, Praha (1933–1957).
7. Kraus, J., Petráčková, V., et al.: Akademický slovník cizích slov (Academic Dictionary of Foreign Words, SCS). Academia, Praha (1999) electronic version, LEDA, Praha.
8. Čermák, F., et al.: Slovník české frazeologie a idiomatiky I-IV (Dictionary of Czech Phraseology and Idioms, SČFI). Academia, Praha (1983).
9. Kilgarriff, A., Rychlý, P., Smrž, P., Tugwell, D.: The Sketch Engine. In: Proceedings of the Eleventh EURALEX International Congress, Lorient, France, Universite de Bretagne-Sud (2004) 105–116.
10. Fillmore, C., Baker, C., Sato, H.: FrameNet as a 'net'. In: Proceedings of Language Resources and Evaluation Conference (LREC 04). Volume vol. 4, 1091-1094., Lisbon, ELRA (2004).
11. Vossen, P., ed.: EuroWordNet: A Multilingual Database with Lexical Semantic Networks for European Languages. Kluwer Academic Publishers, Dordrecht (1998).
12. Maks, I., Martin, W., de Meerseman, H.: RBN Manual. (1999).
13. Niles, I., Pease, A.: Linking Lexicons and Ontologies: Mapping WordNet to the Suggested Upper Merged Ontology. In: Proceedings of the IEEE International Conference on Information and Knowledge Engineering. (2003) 412–416.

Parsing System with Contextual Constraints

Vladimír Kadlec

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

Abstract. The aim of the presented work is to design algorithms and methods for an effective and robust syntactic analysis of natural languages. The algorithms are language-independent, any language with an appropriate grammar can be modeled. The analysis of sentences by the described system is based on context-free grammar for a given language supplemented by context sensitive structures. The internal representation of derivation trees allows to apply contextual constraints, e.g. case agreement fulfillment. The evaluation of semantic actions and contextual constraints helps us to reduce a huge number of derivation trees and we are also able to calculate some new information, which is not contained in the context-free part of the grammar. Also *n*-best trees (according to a tree rank, e.g probability) can be selected. This is an important feature for linguistics developing a grammar by hand.

1 Introduction

Syntactic analysis is a "corner-stone" of applications for automated processing of texts in natural languages. Any machine translation application, an automatic grammar checker or information retrieval system must be capable of understanding the structure of a sentence. Recognition of the sentence structure is called *parsing*.

The analysis of sentences by the described system is based on context-free grammar for the given language. Context-free parsing techniques are well suited to be incorporated into real-world nature language processing systems because of their time efficiency and low memory requirements. Though, it is known that some natural language phenomena cannot be handled with the context-free grammar formalism, researchers often use the context-free backbone as the core of their grammar formalism and enhance it with context sensitive feature structures (e.g. [1]).

2 System Overview

Described system consists of several independent modules. The modular design makes the system easily extensible and rather flexible. Figure 1 shows the data flow through the system.

There are several inputs to the system:

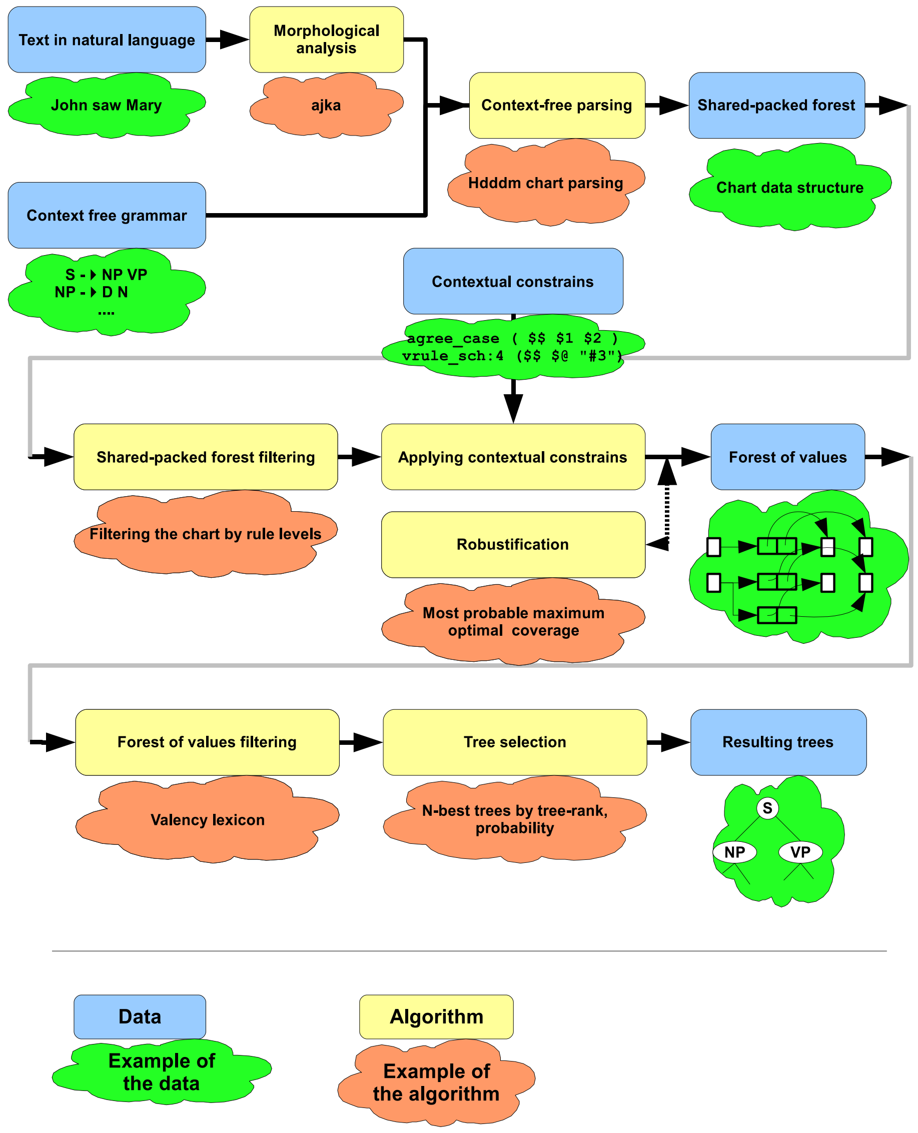


Fig. 1. Parsing System with Contextual Constraints

– A sentence in a natural language.
– Context-free (CF) grammar.
– Semantic actions and contextual constraints for grammar rules.

Words in the input sentence can be optionally tagged. If they are not tagged, then the internal "tagger" is used. The notation of "tagger" is not correct here, because we leave ambiguities in tags. For the Czech language, morphological analyzer `ajka` [2] is used to create tags. For other languages the tags are usually read from an static lexicon. These tags are stored as "values" (see below) for every word. The terminals (or sometimes called pre-terminals) for given context-free grammar are created by simplification the tags, e.g. using only word category as a terminal.

Once the terminals are created the context-free parsing algorithm is run. This algorithm produces all possible derivation trees at the output with respect to the input sentence and input grammar. All these derivation trees are stored in a data structure based on shared-packed forest [3]. Because a chart parser is used in our system [4], the derivation trees are stored as a chart data structure [5,6] directly. Any context-free parsing algorithm could be used, the modularity of the system allows us compare the effectiveness of these algorithms easily [7].

All derivation trees created in the previous step can be filtered by some basic filter, that cuts some trees off. In this step only basic filtering "compatible" with the shared-packed forest data structure is allowed. E.g. only a whole sub-forests can be removed. The example of such filtration is in Section 3.

The next step is application of contextual constraints and semantic actions. In this step a new data structure is created, a "forest of values". The forest of values is created by a bottom-up recursive run of semantic actions, see Section 4.

If the input sentence cannot be generated by the input grammar, i.e. there is no derivation tree at the output of the context-free parsing algorithm, the system offers a robust module. In this case, the contextual constraints and semantic actions are applied on every derivation sub-tree in the shared-packed forest. Then the robust algorithm presented in [8] is used to get the derivation tree(s).

The resulting forest of values can be further filtered by constraints, that work with the whole forest, not only with local values. The example of this global filtering is usage lexicon of verb valencies VerbaLex, see [9].

In the end, the derivation trees are generated from the filtered forest of values. Only one or several "best" derivation trees can be created, with respect to the ranking function, e.g. a probability of the tree could be used as one input to the ranking function.

In the following sections, the above ideas are described in more detail.

3 Shared-Packed Forest filtering

This filtering is used only to remove some data from the structure, no new information is added. This technique is a step preceding the filtering by contextual constraints, see Figure 1 in Section 2 for an overview of the whole system. The resulting shared-packed forest is always sub-forest of the original. That means that only simple transformations such as removing a node is done here.

3.1 Filtering by rule levels

One of possible filtering the shared-packed forest is a local (with respect to the node of the forest) filtering based on what we call "rule levels". The rule level is a function, that assigns a natural number to every grammar rule. In the following the term "a rule level of a grammar rule" denotes the resulting number of application this function to the rule.

The idea is, that for some grammar rules: if the specific grammar rule succeeds during the parse process, then an application of some other rule (with the same non-terminal on the right hand side of the rule) is wrong. To be more precise, if the specific grammar rule covers the same input as some other grammar rule beginning with the same non-terminal, then the rule with lower rule level is refused.

The chart structure [5] represents the shared-packed forest. So the filtering method is described in terms of the chart parsing: If there are edges $[A \rightarrow \bullet\alpha\bullet, i, j]$ and $[A \rightarrow \bullet\beta\bullet, i, j]$ in the chart, then delete the edge with the grammar rule with the lower rule level. If the edges have the same rule level, keep them both in the chart. Figure 2 shows an example of such rules, $w_1, w_2, \ldots, w_6$ represent the input sentence.

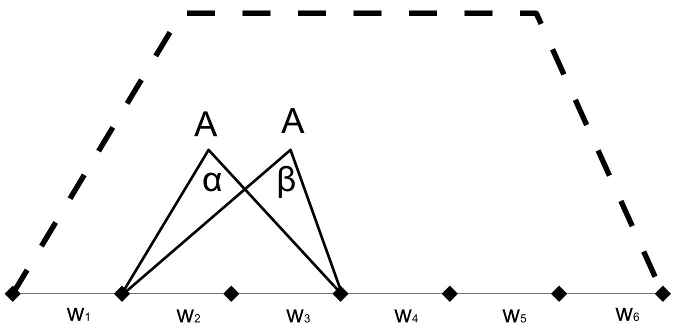


Fig. 2. Filtering by rule levels. Two sub-forests with grammar rules $A \rightarrow \alpha$ and $A \rightarrow \beta$ in their roots. One of them is filtered out, if these rules has a different rule level set.

Notice that this kind of filtering is different from a probability of the grammar rule. The presented method is local to the specific node in the shared-packed forest. By default all grammar rules have the same rule level. The rule levels are set by hand and only in very specific cases. Actually, only one rule in our grammar for Czech [10] has non-default rule level. Only small number of experiments were performed, because this method is new to our system.

4 Contextual Constraints and Semantic Actions

Our main problem with the context-free (CF) parsing is, that there are too many derivation trees for a given input sentence. The contextual constraints are mainly used to prune incorrect derivation trees from the CF parsing result. Also some additional information can be computed by these constraints, that is why we also call them "semantic actions". In the following the term "contextual constraint" has the same meaning as the term "semantic action". Our algorithms for CF parsing generates the chart structure, thus we use the word "chart" to denote "a result of the CF parsing".

See Figure 1 to have a better view, in which part of the parsing system the constraints are computed.

The contextual constraints (or actions) defined in the grammar can be divided into four groups:

1. rule-tied actions
2. case agreement constraints
3. post-processing actions
4. actions based on derivation tree

The example of a rule-tied action is a rule-based probability estimation. Case agreement constraints serve as chart pruning actions. The case agreement constraints represent the functional constraints, whose processing can be interleaved with that of phrasal constraints.

The post-processing actions are not triggered until the chart is already completed. Actions on this level are used mainly for computation of analysis probabilities for a particular input sentence and particular analysis. Some such computations (e.g. verb valency probability) demand exponential resources for computation over the whole chart structure. This problem is solved by splitting the calculation process into the pruning part (run on the level of post-processing actions) and the reordering part, that is postponed until the actions based on derivation tree.

The actions that do not need to work with the whole chart structure are run after the best or *n* most probable derivation trees have been selected. These actions are used, for example, for determination of possible verb valencies within the input sentence, which can produce a new ordering of the selected trees, or for the logical analysis of the sentence [11].

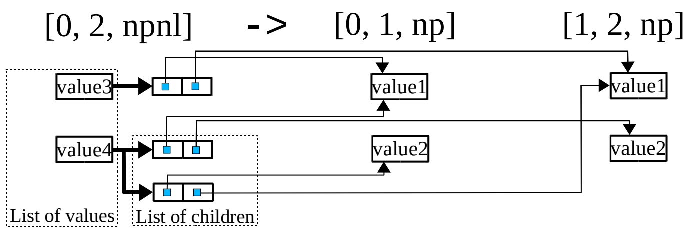


Fig. 3. Example of the forest of values.

4.1 Representation of values

It was shown that parsing is in general NP-complete if grammars are allowed to have agreement features [12]. But the pruning constraints in our system are weaker than for example general feature structures [13].

We allow a a node in the derivation tree to have only limited number of features. We call the features "values", because they rise as results of our semantic actions. E.g. the number of values for noun groups in our system is at most 56. To compute the values, we build a new structure, a forest of values, instead of pruning or extending the original chart.

The forest of values is computed by the depth-first walk through the chart structure. The chart can be viewed as oriented graph. Every edge in the chart is passed only once, the edge can generate at most one node in the new forest of values.

The value is computed as a result of the semantic action – for the grammar rule given by the current edge. The parameters for the semantic action are filled from the values on lower level, "lower" with respect to the derivation tree, i.e. closer to the leaves of the tree. So also arguments of the semantic action are limited by the limit (e.g. 56 possibilities in our case). Because there could be more than one derivation tree containing the current edge, all possible combination of values are passed to the semantic action. The worst-case time complexity for one node in the forest of values is therefore 56^{δ}, where δ is the length of the longest right-hand side grammar rule. Notice that this complexity is independent of the number of words in input sentence.

The values in the forest of values are linked with the edges backwards. An edge contains a single linked list of its values. Each value holds a single linked list of its children. The child is one dimensional array of values. This array represents one combination of values that leads to the parent value. Notice that there can be more combinations of values that lead to the same value. The i-th cell of the array contains a reference to a value from i-th symbol on the RHS of the corresponding grammar rule. The i-th symbol has not to be used to compute the parent value. We use only reference to the edge from such unused cell.

The Figure 3 shows an example representing the rule $npnl \rightarrow np\ np$ and containing three edges ($[0, 2, npnl \rightarrow \bullet np\ np \bullet]$, $[0, 1, np \rightarrow \bullet \alpha \bullet]$, $[1, 2, np \rightarrow \bullet \beta \bullet]$). The right hand sides of each rule are not shown in the figure, they play no role here. $np \rightarrow \alpha$ and $np \rightarrow \beta$ are some rules in the input grammar.

Each np edge contains two values, *value1* and *value2*. This gives us four possible combinations. The semantic action computes from combinations *value1* $\times$ *value2* and *value2* $\times$ *value1* the same value *value4*. The combination *value2* $\times$ *value2* was classified as incorrect (by the action – contextual constraint), so it is not here.

4.2 Generation of a grammar with values

It is possible to create CF grammar, without our contextual constraints, which generates the same derivation trees as the CF grammar supplemented by the constraints. In the following, a method, that for the given input generates a such CF grammar without values, is provided. This allows us to compare our system, that is able to evaluate the constraints, with other systems able to work only with "pure" CF grammars.

We use the following procedure for every inactive edge $[i, j, A \rightarrow X_1 X_2 ... X_n \bullet]$ in the chart:

- for every value v in the edge, we generate the rule: $A \rightarrow A\_value$, where $value$ is an unique textual representation of the value v.
- for every child of the value v, we generate the rule: $A\_value \rightarrow X_1^{'} X_2^{'} ... X_n^{'}$, where $X\_i^{'}$ is:
  - $X_i\_value_i$ if a value $value_i$ from i-th non-terminal is used to compute the value v.
  - X_i otherwise.

Duplicate rules are removed.

Why are the actions and semantic constraints used when they can be replaced by a grammar with values? There are three main reasons. First of all, the grammar with values for all possible inputs would be extremely large, even if the domain range is limited, e.g. by 56 in our case. Secondly, the actions can be easily changed and debugged when computed separately. The third reason is that some of our experiments use semantic actions with unlimited domain range and these actions cannot be substituted by the grammar.

5 Conclusions

In this work, the language independent parsing system is presented. It is based on context-free parser supplemented by contextual constraints and semantic actions.

The evaluation of semantic actions and contextual constraints helps us to reduce a huge number of derivation trees and we are also able to calculate some new information which is not covered in the context-free part of the grammar. The dependency graph or filtering by valency lexicon are examples of such information. The experiments with dependency graphs are at the beginning. But even for some kinds of short non-projective sentences, the correct dependencies can be generated within our approach as well.

All described algorithms are integrated in the parsing system `synt` [10,14]. Future research is aimed at the experiments with verb valences and lexicon of verb valencies for the Czech VerbaLex.

Acknowledgements This work has been partly supported by the Academy of Sciences of Czech Republic under the project 1ET100300414 and by the Ministry of Education of CR within the Center of basic research LC536 and by the Czech Science Foundation under the project 201/05/2781.

References

1. Neidle, C.: Lexical-Functional Grammar (LFG). In Asher, R.E., ed.: Encyclopedia of Language and Linguistics. Volume 3. Pergamon Press, Oxford (1994) 2147–2153.
2. Sedláček, R.: Morphemic Analyser for Czech. Ph.D. thesis, Faculty of Informatics, Masaryk University, Brno, Czech Republic (2005).
3. Tomita, M.: Efficient Parsing for Natural Languages: A Fast Algorithm for Practical Systems. Kluwer Academic Publishers, Boston, MA (1986).
4. Kadlec, V., Smrž, P.: How many dots are really needed for head-driven chart parsing? In: Proceedings of SOFSEM 2006, Czech Republic, Springer-Verlag (2006) 483–492.
5. Kay, M.: Algorithm schemata and data structures in syntactic processing. In: Report CSL-80-12, Palo Alto, California, Xerox PARC (1989).
6. Earley, J.: An efficient context-free parsing algorithm. In: Communications of the ACM. Volume 13. (1970) 94–102.
7. Kadlec, V., Smrž, P.: PACE - parser comparison and evaluation. In: Proceedings of the 8th International Workshop on Parsing Technologies, IWPT 2003, Le Chesnay Cedex, France, INRIA, Domaine de Voluceau, Rocquencourt (2003) 211–212.
8. Kadlec, V., Ailomaa, M., Chappelier, J.C., Rajman, M.: Robust stochastic parsing using optimal maximum coverage. In: Proceedings of The International Conference Recent Advances In Natural Language Processing (RANLP) 2005, Shoumen, Bulgaria, INCOMA (2005) 258–263.
9. Hlaváčková, D., Horák, A., Kadlec, V.: Exploitation of the Verbalex verb valency lexicon in the syntactic analysis of Czech. In: Proceedings of Text, Speech and Dialogue 2006, Brno, Czech Republic, Springer-Verlag (2006) 85–92.
10. Horák, A., Kadlec, V.: New Meta-grammar Constructs in Czech Language Parser synt. In: Proceedings of Text, Speech and Dialogue 2005, Karlovy Vary, Czech Republic, Springer-Verlag (2005) 85–92.
11. Horák, A.: Analysis of Knowledge in Sentences. Ph.D. thesis, Faculty of Informatics, Masaryk University, Brno, Czech Republic (2002).
12. Barton, G.E., Berwick, R.C., Ristad, E.S.: Computational complexity and natural language. MIT Press, Cambridge, Massachusetts (1987).
13. Kay, M.: Parsing in functional unification grammar. In: Natural Language Parsing, England, Cambridge (1985) 251–278.
14. Horák, A., Kadlec, V.: Platform for Full-Syntax Grammar Development Using Meta-grammar Constructs. In: Proceedings of the 20th Pacific Asia Conference on Language, Information and Computation, Beijing, China, Tsinghua University Press (2006) 311–318.

Morphological Analysis of Law Texts

Karel Pala, Pavel Rychlý, and Pavel Šmerk

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

Abstract. In the paper we explore the morphology of the Czech law texts including Constitution, acts, public notices and court judgements which form a huge textual database. As many texts from small domains, the used language is partially restricted and in relevant aspects also different from general Czech.
The paper presents first results of the morphological analysis of Czech law texts and their conversion to the specific formats. Partly, the partial syntactic analysis has been performed as well.

1 Introduction

In the paper we describe the first results of the new project whose final goal is to build an electronic dictionary of Czech law terms. We start with a legal database Lexis which presently includes approx. 50,000 Czech law documents ranging from the beginning of Czechoslovakia in 1918 to present days. It also includes court judgements, main representative law textbooks and law reports. All the texts exist in electronic form.

1.1 Pilot project

As a pilot project we have decided to analyse the current version of the Penal Code of the Czech Republic. It is one of the biggest law documents containing almost 36,000 word forms. The overall characteristic of the document can be found in Table 1.

The task is to process the document by the Czech morphological analyser (lemmatizer) Ajka in such a way that for each word form in the source text a morphological information in the form of morphological tags is obtained. Thus we get information to what parts of speech the word forms belong, and, for instance, for nouns also grammatical categories like gender, number and case. Each word form in the document is associated with its respective lemma as well. In the highly inflectional language like Czech all this information is relevant for the further analysis of law terms. The results of the morphological analysis and lemmatizations are transformed into a special format which is described below.

Table 1. The overall characteristic of the Penal Code of the Czech Republic

Number of	
word forms (tokens)	35,893
numbers	2,647
punctuation marks	9,135
tokens total	47,865
different word forms (types)	5,019
different numbers	467
different punctuation marks	12
types total	5,019

2 Morphological Analysis

We have used several simple scripts to create what is called vertical file from the source text. It is a plain text file without any formatting (word-processing options). Words are written in a column, i.e. each line contains one word, number or punctuation. Optional annotation is on the same line and the respective words are divided by the tabulator character. The first step uses only word forms from the source text. The vertical file serves as an input text for many corpus processing tools like CQP [1] and Manatee [2].

In the next step, we processed the vertical file with the morphological analyser Ajka [3]. It is a tool exploited for annotating and lemmatizing general Czech texts, however, the processing law texts requires modifications, e.g. enriching the list of stems of Ajka. The programme yields all possible combinations of lemma and morphological tags for each Czech word form. In the following example of the Ajka output the tag **k1gFnSc1** means: part of speech (**k**) = noun (**1**), gender (**g**) = female (**F**), number (**n**) = singular (**S**) and case (**c**) = first (nominative) (**1**), tags beginning with **k2** are adjectives, **k3** – pronouns, **k5** – verbs and **k7** – prepositions.

Příprava	<l>příprava <c>k1gFnSc1 (preparation)
k	<l>k <c>k7c3 (to)
trestnému	<l>trestný <c>k2eAgMnSc3d1 <c>k2eAgInSc3d1 <c>k2eAgNnSc3d1 (criminal)
činu	<l>čin <c>k1gInSc3 <c>k1gInSc6 <c>k1gInSc2 <l>čina <c>k1gFnSc4 (act)
je	<l>být <c>k5eAaImIp3nSrDaI <l>on <c>k3p3gMnPc4xP <c>k3p3gInPc4xP <c>k3p3gNnSc4xP <c>k3p3gNnPc4xP <c>k3p3gFnPc4xP <l>je <c>k0 (is)
trestná	<l>trestný <c>k2eAgFnSc1d1 <c>k2eAgFnSc5d1 <c>k2eAgNnPc1d1 <c>k2eAgNnPc4d1 <c>k2eAgNnPc5d1 (criminal)

As one can see, many word forms are ambiguous: there are more than one possible tag or even lemma for a given word form. In the analysed document, 76 % of word forms are ambiguous, more than 42 % of word forms have more than one possible lemma and average number of tags for an ambiguous word form is 6.75.

We have used part-of-speech tagger Desamb [4] to disambiguate such word forms. The output of the Desamb tool contains only the most probable lemma/tag for each word form. Table 2 contains output of Desamb for the input text above.

Table 2. The document in vertical format with morphological annotation (after disambiguation)

Příprava	příprava	k1gFnSc1
k	k	k7c3
trestnému	trestný	k2eAgInSc3d1
činu	čin	k1gInSc3
je	být	k5eAaImIp3nS
trestná	trestný	k2eAgFnSc1d1
podle	podle	k7c2
trestní	trestní	k2eAgFnSc2d1
sazby	sazba	k1gFnSc2
stanovené	stanovený	k2eAgFnSc2d1
na	na	k7c4
trestný	trestný	k2eAgInSc4d1
čin	čin	k1gInSc4

The annotated version of the document contains 2,560 different lemmas. Frequencies of each part of speech are in Table 3.

Table 3. Frequencies of part of speech in the document

Part of Speech	Count
k1 – noun	12884
k2 – adjective	4634
k3 – pronoun	2252
k4 – numeral	1028
k5 – verb	4504
k6 – adverb	933
k7 – preposition	3600
k8 – conjunction	3764

3 Noun Groups

For the recognition of the noun groups we have used the partial syntactic analyzer for Czech DIS/VADIS [5] at first. Unfortunately, DIS/VADIS presently does not contain rules which can recognize genitival and coordinate structures because during the development of DIS/VADIS these rules were found too erroneous (overgenerating) when applied to an unrestricted text. However, there are plenty of such structures in the law texts and overgenerating is not a problem here because the results will be checked manually.

Moreover, the partial syntactic analyzer DIS/VADIS has one more disadvantage: it is written in Prolog which implies that the recognition process is rather slow. Therefore we have rewritten the rules for noun groups to Perl 5 regular expressions (which have nontrivial backtracking capabilities) and added the rules for genitival and coordinate structures and some adverbials common to the law texts which also were not recognized by DIS/VADIS (e.g. *zvlášť* (exceedingly), *zjevně* (evidently) etc.).

For each noun group found in the law texts we determine its:

1. base form (nominative singular),
2. head
3. for nouns in genitive groups also their part.

For example for the noun group *dalším páchání trestné činnosti (subsequent commission of criminal activity, dative)* we get:

1. *další páchání trestné činnosti*
2. *páchání*
3. *další páchání*

We can recognize 8,594 noun groups counting repeating occurencies, 3,992 different noun groups. Table 4 lists several most frequent noun groups (since there are problems with finding the correct English equivalent terms we do not offer them here). Table 5 presents the most frequent part-of-speech patterns of the recognized noun groups.

Table 4. The most frequent noun groups

Noun Group	Count
odnětím svobody	492
peněžitým trestem	139
jeden rok	123
trestný čin	79
odnětí svobody	76
účinnosti dne	65
zákazem činnosti	64
trestného činu	58
velkého rozsahu	49
závažný následek	47
zvlášť závažný následek	46

Table 5. The most frequent POS patterns

Part of Speech Patterns	Count
k2 k1gI	1526
k2 k1gF	1127
k1gN k1gF	769
k2 k1gN	469
k1gI k1gN	210
k1gN k1gI	203
k1gI k1gF	193
k1gF k1gI	177
k1gF k1gN	171
k1gF k1gF	164

4 Verb List

Though law terms typically consist of the nouns, noun groups and other nominal constructions we also have paid attention to the verbs found in the whole database of the 50,000 law documents. The reason for this comes from the fact that verbs on one hand do not display strictly terminological nature but on the other they are relational elements linking the terminological nouns and noun groups together. This can be captured by the surface and deep verb valency frames [6] of the verbs occuring in the law documents. We are not aware of any attempt to describe the valency frames of the verbs coming from law texts. Presently the verb list comprises 15,110 items, particularly 10,190 infinitives and 4,920 participles (which are mostly the passive ones). The list has been processed by the morphological analyzer Ajka [3] as a result we have obtained the list of 914 items that were not recognized by Ajka tool. The structure of this list shows that at least three types of the non-recognized items can be observed:

1. erroneous forms caused by typing errors, they can be corrected, e. g. *cítít (feel),*
2. the verbs that Ajka does not know, i. e. the ones that do not appear in the Ajka's list of stems. Typically, they display a terminological character and they should be added to the Ajka's stem list, e. g. *derogovat (derogate)*. They will enrich the list of (Czech) stems and their law meanings constitute a terminological subset of verbs,
3. erroneous forms that cannot be corrected without correcting the whole paragraph of a law document.

The next step is to add the non-recognized verbs to Ajka's list of the verb stems and to make an intersection with our existing database Verbalex [6] containing presently about 11,306 (general) Czech verbs.

5 Conclusion

We have presented the preliminary results of the computational analysis of Czech law documents, or more precisely, their samples. On one hand we have used the already existing tools such as Ajka or DIS/VADIS, on the other hand we have modified respectively them for the purpose of the present task. As a result we can enrich them with regard to the law language but, more importantly, we have obtained basic knowledge about the grammatical structure of the law texts (law terminology) and in this way we are prepared to continue our exploration of the contexts in which law terms occur in the law documents. The knowledge of such contexts is a necessary condition for a deeper understanding of how law terminology works and how it can be made more consistent. As an application we hope to obtain the basic rules for intelligent searching law documents. A tool based on such rules can serve to judges, attorneys and experts in creating new law documents.

Acknowledgements

This work has been partly supported by the Academy of Sciences of the Czech Republic under the projects 407/07/0679 and by the Ministry of Education of the Czech Republic within the Centre of basic research LC536.

References

1. Schulze, B.M., Christ, O.: The CQP User's Manual. (1996).
2. Rychlý, P.: Corpus Managers and their Effective Implementation. Ph.D. thesis, Faculty of Informatics, Masaryk University (2000).
3. Sedláček, R.: Morphemic Analyser for Czech. PhD thesis, Masaryk University (2005).
4. Šmerk, P.: Towards Morphological Disambiguation of Czech. Ph.D. thesis proposals, Faculty of Informatics, Masaryk University (2007).
5. Žáčková, E.: Partial Syntactic Analysis of Czech. Ph.D. thesis, Faculty of Informatics, Masaryk University (2002).
6. Horák, A., Hlaváčková, D.: VerbaLex – New Comprehensive Lexicon of Verb Valencies for Czech. In: Computer Treatment of Slavic and East European Languages, Third International Seminar, Bratislava, VEDA (2005) 107–115.

Part II

Semantic Analysis

The Learning and Question Answering Modes in the Dolphin System for the Transparent Intensional Logic

Andrej Gardoň and Aleš Horák

Faculty of Informatics, Masaryk University Brno
Botanická 68a, 602 00 Brno, Czech Republic
`xgardon@fi.muni.cz`, `hales@fi.muni.cz`
`http://nlp.fi.muni.cz/projects/dolphin`

Abstract. In this paper, we present the two modes of operation of the developed Dolphin system, which implements a knowledge base for the Transparent Intensional Logic formalism.
We offer several examples of the system input with detailed explanation of how these inputs are parsed and stored in the system and how the system reuses the knowledge base information for answering simple questions offering a kind of simple inference.

Key words: Dolphin, transparent intensional logic, TIL, knowledge base

1 Introduction

The transparent intensional logic (TIL) is a higher order typed logical system designed for representing meaning of human language in a computer. In comparison with traditional logic systems, TIL is able to distinguish so called intensions, i.e. entities dependent on the reference possible world and time moment, and extensions. Using the higher order types, TIL allows us to express *attitudes*[1] to other natural languages objects in a declarative manner. For more detailed information about TIL you can see [1,2].

In the following text, we will present details of an ongoing project called Dolphin, which is an implementation of an efficient knowledge base built over the TIL formalism. The basic ideas of Dolphin have been published in [3].

2 The TIL Knowledge Base Architecture

The Dolphin system is designed to process the output of the syntactic parser `synt` [2,4], which is able to produce syntactic trees as well as logical representations of input sentences in the form of TIL constructions. In Dolphin, the constructions are parsed and stored in the knowledge base (KB), which takes the

[1] e.g. *Tom believes that Peter says that the earth is flat.*

form of a semantic network. The system is then able to process the facts from KB in the way allowing to answer *yes/no* questions and questions about firmly established objects relations in the form of *The apple is red. The cube is red. What red objects do you know?*. Although current version of the system does not support all TIL capabilities in full breadth, we hope the design of it will make it easy to add them in the future.

2.1 How the Dolphin Database Works

The Basic Dolphin idea is a separation of the *language layer* from the *logical layer*. Human words are just a way how to describe some objects. There are many languages but all of them work over the same set of objects. Therefore Dolphin works with this set and is ready to understand everything what his teacher teaches him. This process lies in transforming teacher words to the KB objects mentioned above. Here `synt` plays its role as it produces TIL transcription of sentence. As a young child who gets in touch with word *apple* for the first time, Dolphin after obtaining *apple* on its input takes its TIL description and creates new object in its object layer – let it be object #1. Next time we mention this particular *apple*, Dolphin realizes object #1 in its KB and whatever is told about it is appended to this object.

2.2 The Role of the Language Layer

The linkage from the apple and the object #1 is driven by the language layer. The word *apple* is stored in the appropriate language file and is connected to the object #1. If the system is switched to the *question answering mode* (see the Section 2.3), whenever the word *apple* appears, it is replaced by object #1 for the purpose of the logical inference. The current implementation of the language layer supports multiple languages but the rest of the system works, at the moment, with one selected language. The reason for this is that the language learning needs to employ specific inference. The language layer also offers techniques for synonyms and homonyms but again complex inference is needed to use it. The implementation of the language layer uses an adapted library of special B-trees and thus provides effective transcription of a word to the equivalent object.

2.3 The Input Sentence Processing

Let us take an example input sentence (in Czech, as this is the current input language of `synt`):

Jablko je červené. (An apple is red)

and its corresponding TIL transcription:

$$\lambda w_1 \lambda t_2 (\exists i_3)([{}^0\text{jablko-0}_{w_1 t_2}, i_3] \,\wedge\, [{}^0\text{červený-2}_{w_1 t_2}, i_3]) \ldots \pi$$

At first, variables are identified and single applications are isolated. Variables that are not covered by λ-abstraction are replaced by new constants named after the variables. The λ-abstracted variables are currently handled in a simplified way – when the λ-abstraction is used without quantification, the abstracted variables usually go over the ω (possible worlds) or τ (time moments) types. In case of an ω-variable, the Dolphin's world $w_{Dolphin}$ is assigned to it. All time variables (of type τ) are, in the current version, replaced by an object representing a "general time independent object," so as we can see the current version does not support time processing at all. This feature will be the main goal of our future work. Other quantified variables are initialized differently according to the system running mode. There are two running modes currently available – the *learning mode* and the *question answering mode*.

The Learning Mode processes and stores new facts and it is activated by a full stop mark at the end of a sentence. In this mode, each uninitialized variable is replaced by a new object named after the variable. If we want to add new facts about an object previously mentioned in the conversation with Dolphin, we have to stress this with the demonstrative pronoun (*ten*, *ta*, *to* – in Czech this corresponds to the definite article in English). The object is then marked with an exclamation mark (!) and is looked up in the knowledge base and the variable is replaced by the stored object. For example if we want to add the fact that

To jablko je červené. (The apple is red.)

the *apple* is analysed as $^{0}\mathsf{!jablko\text{-}0}/(oi)_{\tau\omega}$ and then found as the previously stored object #23 and the construction would thus contain object $^{0}\#23$ instead of the i_3 variable displayed above. Currently, the system supports only the existential quantification since the universal quantification defines new inference rules and the complex inference has not been implemented so far.

We may mentioned the way, how possible fact conflicts are handled – if the input fact assigns something to an existing KB object and the fact is in conflict with the KB content then an error message is raised. For demonstration let us have a sentence

The apple is red.

stored in KB. Now we would like to store the sentence

The apple is not red.

This raises an error message and the second sentence is not stored. Today there is no tool for saying to Dolphin:

A fact in KB is wrong, I have the correct one.

so whatever is stored in KB will remain unchanged until a reset of the whole database.

The Question Answering Mode works similarly to the learning mode but there is no unification of free variables with new objects. Instead, the first application containing an uninitialized variable suggests a way how to unify this variable with a particular value. If we take our example sentence as a question

Je nějaké jablko červené? (Is there any red apple?)

first application containing free variable i_3 (variables w_1 and t_2 are initialized as described above) will be

$$[{}^0\mathsf{jablko\text{-}0}_{w_1 t_2}, i_3]$$

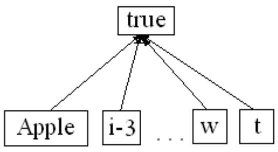
Thanks to this application, i_3 is unified with an object that is stored in Dolphin and is an Apple (is linked to the class construction of apples). The following application (saying that the object is *red*) posts a second requirement to check. If the object unified with i_3 is not Red (there is no relation between the unified object and the class of red objects), the system returns to the state when i_3 was initialized and tries another possibility. If all possible ways were checked and there is no object that is Apple and Red concurrently the answer *NO* is returned. In case all the requirements are fulfilled, the answer *YES* is returned with the selected i_3 value. Again there is no universal quantification feature yet, but simple questions such as

Jaké červené objekty znáš? (What red objects do you know?)

can be processed correctly.

2.4 How the Objects are Stored

Objects are essential elements of logical analysis and the TIL construction processing. In the Dolphin knowledge base, all data are objects related with each other. It does not matter if we store an individual or a function in KB, we are still working with one object. For example, the class of apples (Apple) is represented as the characteristic function of the class, i.e. as a (world and time dependent) function which returns an o-object (boolean *true*/*false*) for each ι-object (an individual) given as its argument. Each such ι-object is stored separately from Apple, of course.



There is an information that Apple and the ι-object i_3 are in relation and this information is shared among all participants. Each object is represented as a separate file. This can be considered as very ineffective while each time we need an object we have to read a separate file from disc. In fact, there is no solution that does not use disc reading as there is so many objects and absolutely no chance to store them all in operating memory. On the other hand simple file for each object offers some benefits. In the future, a specialized file system technology can be developed or archiving methods can be incorporated in the storage process without the database functionality limitation or reimplementation.

3 A Complex Example

Let us take three example sentences:

1. *Toto je jablko. (This is an apple.)*
2. *Tato kostka je červená. (This cube is red.)*
3. *To jablko je červené. (The apple is red.)*

and their TIL transcriptions provided as inputs to Dolphin, step by step:

$$\lambda w_1 \lambda t_2 ([{}^0\mathsf{jablko\text{-}0}_{w_1 t_2}, \mathit{Toto}]) \ldots \pi$$

The word *toto* (this) makes the resulting construction an open construction (with the *free* variable *Toto*) which needs to receive so called *pragmatic anchor*.[2] This input creates a new object in the knowledge base and Dolphin prints the assigned object number:

```
> Toto je jablko. (This is an apple.)
> stored as object 6.
```

The second sentence

```
> Tato kostka je červená. (This cube is red.)
```

with the corresponding TIL construction

$$\lambda w_1 \lambda t_2 ([{}^0\mathsf{!kostka\text{-}0}_{w_1 t_2}, i_3] \ \wedge \ [{}^0\mathsf{červený\text{-}2}_{w_1 t_2}, i_3]) \ldots \pi$$

where the variable w_1 is unified with object #2 (representing the Dolphin's world), the variable t_2 is unified with object #3 (representing the General time object) and the variable i_3 is now initialized with a new object of type ι (let it be #7), since no previously mentioned object from the class *kostka* was found.

Each trivialization asks the language layer whether it knows the word. If the word is not found, it is stored and a new object is created with connection to this word.

So $\wedge$ (AND) in our transcription causes that search in language layer is performed and object #10 is returned (we suppose that AND was previously stored).

[2] see [2, the Section 5.2.4] or [5, the Section 7.1]

Applications are represented as relations among objects that are participating in the application. Thus in Dolphin, the partial application

$$[{}^0\text{červený-2}/(oi)_{\tau\omega} w_1]$$

is represented as a relation between *červený* and the object of the variable w_1. The information about the relation is stored both in object *červený* and w_1 object and of course in the final object – the result of this application is a new $(o\iota)\tau$-object (a *chronology* of the class of objects which are červený/red) which is than applied on the general time object. The final object of the application

$$[{}^0\text{červený-2}_{w_1 t_2}, i_3] \ldots o$$

is then created as an o-object (a truth value) and in the learning mode the object receives the *true* value.

The third example sentence is stored similarly as the second one with the i_3 variable definition difference.

```
> To jablko je červené. (The apple is red.)
```

TIL transcription

$$\lambda w_1 \lambda t_2 ([{}^0\text{!jablko-0}_{w_1 t_2}, {}^0\#6] \wedge [{}^0\text{červený-2}_{w_1 t_2}, {}^0\#6]) \ldots \pi$$

Semantic network for three sentences is displayed in the Figure 1. Note that there are many objects with *true* value in the network. This is because each application leads to a unique object and if this object is not yet in KB, it is created. Thanks to this, the λ-abstraction can be done in a straightforward way. It is worth saying that negation is a special function over the o-object that does not create a new object but it replaces the existing *true* value with *false*. If we ask the question

Je to jabko červené? (Is the apple red?)

TIL transcription is in form of a match:

$$x \ldots o : ([{}^0\text{!jablko-0}_{w_{Dolphin} t_{now}}, {}^0\#6] \wedge [{}^0\text{červený-2}_{w_{Dolphin} t_{now}}, {}^0\#6]) \ldots o$$

Basically it asks whether the previously mentioned object #6 is in relation with Apple and Red concurrently. It is easy to answer as we have this information in our knowledge base already. System just obtains the result of the applications on the right side of the match (common applications of objects) and checks whether final object of the right side is *true*. In this case, the answer is *yes* but what happens if we have the question

Je to jablko zelené? (Is the apple green?)

Since we are in the question answering mode, the basic trivialization of *green* will fail as we do not have such object in KB yet. The system does not follow the predicate logic *closed world assumption*, thus the answer is “I DO NOT KNOW” instead of “NO.” The question

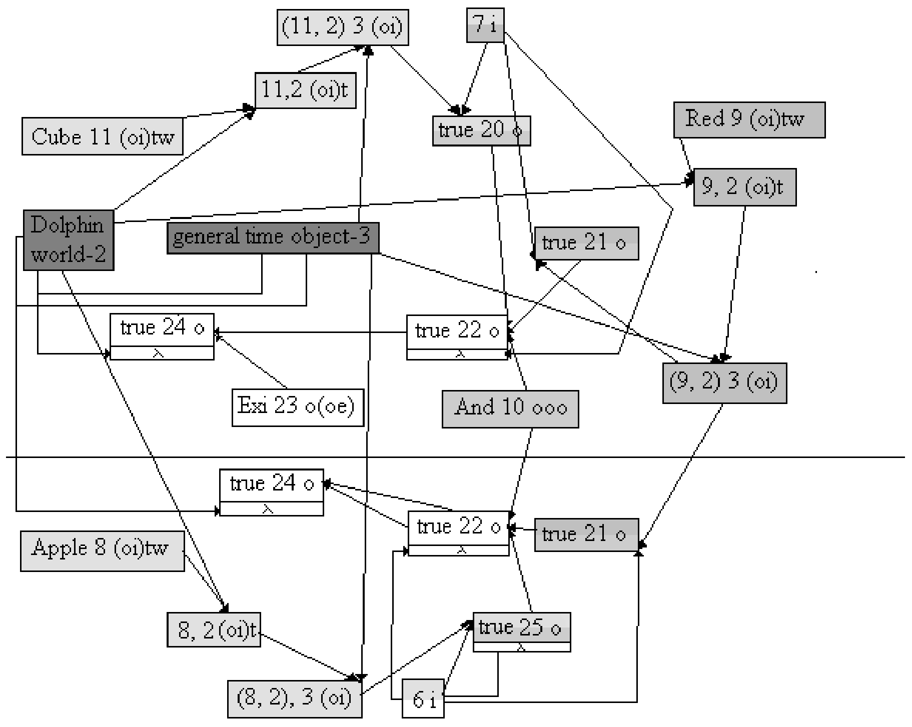


Fig. 1. The semantic network for the 3 example sentences.

Které červené objekty znáš? (What red objects do you know?)

is again analysed as a match

$$s \ldots (oi) : {}^0\text{červený-1}_{w_{Dolphin} t_{now}} \ldots (oi)$$

Here the system evaluates all possible objects which are stored in KB as related to the class *červený* (applied on the Dolphin's world and the general time object) obtaining the class of all red objects known to Dolphin. Thanks to system design, this operation consists in fast searching through the semantic network. As can be seen in the Figure 1, the Red object applied on the Dolphin's world and the General time object ((oi)-object with `(9,2) 3` label) has connections to the *true* value. Now (simplified) it is enough to look what object is at the end of the connection that runs out from the place of the *true* value where the connection from `(9, 2) 3` object ended. In this way we find out that objects #6 and #7 are red.

4 Conclusions and Future Work

We have presented the current state of development of the Dolphin knowledge base for the Transparent Intensional Logic. The system is currently able to store TIL constructions in an efficient semantic network structure with offers basic question answering ability.

The Dolphin system is able to parse and store the `synt` output in the form of TIL constructions, to check the consistence of the database and to answer simple questions. The final aim of the Dolphin development is the full support of working with possible worlds and time moments and the provision of a complex inference tool. The directly following version will be directed to the time span support. The plan is that every mentioned time moment will be represented as a separate object, which will contain contain information about objects that are in relation with it. Through this system a simple fact will have different truth value in various time moments and it will be possible to answer questions like:

What is true in the time moment XXX?

Acknowledgments

This work has been partly supported by the Academy of Sciences of Czech Republic under the project T100300414 and by the Czech Science Foundation under the project 201/05/2781.

References

1. Tichý, P.: The Foundations of Frege's Logic. de Gruyter, Berlin, New York (1988).
2. Horák, A.: The Normal Translation Algorithm in Transparent Intensional Logic for Czech. Ph.D. thesis, Masaryk University, Brno (2002).
3. Gardoň, A., Horák, A.: Dolphin – a Knowledge Base for Transparent Intensional Logic. In: Proceedings of the Seventh International Workshop on Computational Semantics (IWCS-7), Tilburg, The Netherlands (2007) 316–319.
4. Horák, A., Kadlec, V.: New Meta-grammar Constructs in Czech Language Parser synt. In: Proceedings of Text, Speech and Dialogue 2005, Karlovy Vary, Czech Republic, Springer-Verlag (2005) 85–92.
5. Materna, P.: Concepts and Objects. Volume 63 of Acta Philosophica Fennica. The Philosophical Society of Finland, Helsinki (1998).

Functional Programming Based on Transparent Intensional Logic

Nikola Ciprich, Marie Duží, Michal Košinár

VŠB—Technical University Ostrava
Faculty of Electrical Engineering and Computer Science
17. listopadu 15, 708 33, Ostrava-Poruba, Czech Republic
`nikola.ciprich@linuxbox.cz`

Abstract. In the paper we introduce an interpreter of the Transperent Intensional Logic (TIL), namely the TIL-Script language, its design and implementation. TIL is a brainchild of the Czech logician Pavel Tichý and in many ways a revolutionary logical system. The work on the TIL-Script project is being pursued by the team of undergraduate students. The paper we provide is a brief review of the TIL-Script syntax.

Key words: TIL, TIL-Script, Transparent Intensional Logic, syntax, grammar

1 Introduction

1.1 Motivation

Pavel Tichý set his work on TIL in 1970s, and since then he demonstrated its great expressive power, in particular in the area of analysis of natural language processing (see [3,4]). Since hitherto there is no computer implementation of TIL, we decided to create our own and make it available to public.

1.2 Objective

The notational syntax of TIL language of constructions as originally proposed by Tichý is not optimal for computer processing. It contains lots of non-ASCII characters which are difficult to type on a computer; for instance, Greek alphabet is used to denote particular atomic and molecular types, lots of upper indexing, etc. On the other hand, some special types that are useful from the computational point of view are missing (like the type for integers, a special type for time, lists and tuples, etc.). Therefore the changes in syntax and slight modifications in semantics had to be done in order that the code be easy to type and read. Since TIL as a higher-order logic is undecidable, it is also important to create a limited but working inference engine.

2 Constructions

There are six kinds of constructions in TIL, two atomic and four molecular ones. In the TIL-Script code each entry of a construction is terminated by a period.

2.1 Trivialisation

The first atomic construction is *trivialisation*. Given an object, it just returns the object. It could be thought of as compared to a pointer, and its dereference. In TIL, the Trivialisation is denoted by the ^{0} symbol (thus writing ^{0}*Object*). TIL-Script uses the apostrophe (′) symbol, as it's a symbol similar to the original one and easy to type. Thus we have the transcription of ^{0}*Object* into 'Object. Any object, even a construction can be trivialised.

2.2 Variables

Another atomic construction is a variable. From the syntactic point of view, variables in TIL are usually named by lower-case letters; similarly in TIL-Script we can use any name beginning with a lowercase letter and consisting only of letters, numbers and '_' symbol.

2.3 Closures

The third construction is a *Closure* (also a *lambda closure*). It constructs an anonymous function f, that can then be applied to its argument a by using the Composition of the closure with a construction of a. In TIL, the syntax is similar to the lambda-calculus lambda abstraction, using the symbol λ to mark lambda-bound variables. For example, the Closure constructing the successor function can be written as $[\lambda x[^0{+}x^0 1]]$. TIL-Script retains this syntax, just replacing the symbol λ by '\' (and of course using the ′ symbol for trivialisation). Moreover, types have to be specified (more on types see below). So the TIL-Script notation of the *successor* closure is as follows:

```
[\x:Int ['+ x '1]].
```

Creating functions using named closures In TIL-Script, we can also construct a *named* function using Closure. To this end the d**ef** keyword is used; for instance, the above construction of the successor function named as *Succ* is in TIL-Script written as follows[1]:

```
def Succ := [\x:Int ['+ x '1]].
```

2.4 Compositions

The *Composition* construction is an instruction to apply a function to its arguments. There is no notational difference between TIL and TIL-Script concerning a Composition. For instance, here is a Composition of the above Closure with the Trivialisation of the number 5:

```
[[\x:Int ['+ x '1]] '5].
```

[1] Parameter types can also be specified in another way, this will be discussed later

Or, using the pre-defined name of the successor function, we have:

```
['Succ '5].
```

both constructing the number 6.

2.5 Double execution

Some constructions can be used twice over: the *Double Execution* of a construction X, in symbols ${}^2X$, v-constructs what is v-constructed by X. In TIL-Script the upper index ^{2} is replaced by the `^2` notation: `^2X`.

2.6 Partiality

It is important to note that TIL operates with partial functions. Thus a construction can fail to v-construct anything, i.e., it can be v-improper. This is in principle due to a Composition, when a partial function f is applied to an argument a but there is no value of f at a. Thus the Composition of the division function with an argument <x,0> encoded in TIL-Script by ['Div x '0], is v-improper for any x. And so is any Composition of another construction with the former, like, e.g., ['Plus ['Div x '0] '1]. We say that 'partiality is being strictly propagated up'.

Closure never fails to v-construct, it always v-constructs a function, even if it were a function that is undefined at all its arguments like `[\x ['Div x '0]]`.

3 Types

TIL defines four basic types: ι (iota) for the set of individuals (the universe of discourse), o (omicron) for the set of truth values, τ (tau) for the set of times and/or real numbers and ω (omega) for the set of possible worlds. TIL-Script slightly extends and modifies this set of basic types, as the mix of real numbers and times and the absence of integer numbers is not plausible. Therefore the types in TIL-Script are:

- `Bool` or `o` for truth values (o)
- `Indiv` or `i` for individuals (ι)
- `Time` or `t` for time (τ)
- `Real` or `r` for real numbers
- `Int` for integer numbers

Special keyword **`Any`** stands for an unspecified type (α in TIL) that is used to indicate polymorphic function. The frequently used abbreviation of the Composition [[Cw]t] that is used for the extensionalisation of intensions (functions from possible worlds into chronologies of entities of a given type) is replaced by the `C@wt` notation.

3.1 Lists

From the computational point of view, an important type is that of a *list*. A list is a (potentially infinite) sequence of entities and can thus be constructed by a composed TIL construction. However, since lists (or tuples) are frequently used in a computer program, we decided to include list into TIL-Script as a special type. To this end we use the `l`**ist** keyword: `list`(type1 type2 . . .). For example, the list of individuals is declared by `list(Indiv)` or `list(i)`, the list of triples of numbers is declared by `list(Int, Int, Int)`. Lists can also be defined recursively; thus by using `list(list(Int, Indiv))` we declare the list of lists of number-individual pairs. Types are then specified using the slash or colon (in Closures). There are two ways of defining types of function arguments. Either by defining the type of an entity (slash notation), or by defining the range of a variable (the colon notation). To adduce some examples, here are the types of individuals, empirical functions, properties and the range of variable x, respectively:

```
Charles, Tomas/Indiv.
Ostrava/Indiv.
President/(((ii)t)w).
President/(ii)@tw.
Property1/((oi)t)w).
Property2/(oi)@wt.
def Succ1:= [\x:Int ['+ x '1]].
Succ2/(Int Int).
def Succ2:=[\x ['+ x '1]].
```

4 Miscellaneous

4.1 Assignment operator

As an assignment operator, the `l`**et** keyword is used, e.g.: `let` city:='Ostrava. It assigns a value to a variable, and it is used mainly in recursive definitions and in the discourse processing (see [6]).

4.2 Quantifiers

For quantifiers we use the keywords `F`**orAll** (stands for ∀), `E`**xists** (stands for ∃), and `S`**ingle** for singularisers.

4.3 Infix × prefix notation

The first version of TIL-Script will support only prefix notation for all operators, further versions will also support the infix notation.

5 Examples

As an example we now introduce the transcription of the analysis of three agents communication (taken from [6]) into TIL-Script:

```
ind, loc/i.
pred, prof/(oi)@tw.
rel1/(oi(oi)@tw)@tw.
rel2/(oii)@tw.
rel3/(oio@tw)@tw.
prop/o@tw.
constr/*n.
```

Adam to *Cecil*: "Berta is comming. **She** is looking for a parking". 'Inform' message content:

```
\w\t['Comming@wt 'Berta].
```

Discourse variables updates:

```
let ind:='Berta.
let pred:='Coming.
let prop:=\w\t['Coming@wt 'Berta].

\w\t ^2['Sub ind 'she '['Looking_for@wt she 'Parking]].
                                % (is transformed into:)
\w\t['Looking_for@wt 'Berta 'Parking].
```

Discourse variables updates:

```
let rel1:='Looking_for.
let pred:='Parking.
let prop:=\w\t['Looking_for@wt 'Berta 'Parking].
let prof:=\w\t\x['Looking_for@wt x 'Parking].
```

Cecil to *Adam*: "**So** am I". 'Inform' message content:

```
\w\t^2['Sub prof 'so '[so@wt 'Cecil]]. % transforms to:
\w\t['Looking_for@wt 'Cecil 'Parking].
```

Discourse variables updates:

```
let ind:='Cecil.
```

Adam to both: "There is a free parking at *p*1". 'Inform' message content:

```
\w\t[['Free 'Parking]@wt 'p1].
```

Discourse variables Updates:

```
let loc:='p1.
let pred:=['Free 'Parking].
let prop:=\w\t[['Free 'Parking]@wt 'p1].
```

Berta to *Adam*: "What do you mean by free parking?" 'Query' message content:

```
\w\t['Refine@wt '['Free 'Parking]].
```

Adam to *Berta*: "Free parking is a parking and some parts of it are not occupied". 'Reply' message content:

```
'['Free 'Parking] =
'[\w\t\x ['And
  ['Parking@wt x]
  ['Exists y [And
    ['Part_of@wt y x]
    Not ['Occupied@wt y]
  ]
]]].
```

6 Conclusion

In this paper we outlined the syntax of the TIL-Script language that is being developed within the project "Logic and Artificial intelligence for Multi-Agent Systems". TIL-Script is a FIPA compliant language in which the content of FCA messages is encoded. Agents communicate by messaging in TIL-Script.

In the paper we illustrated how smooth and natural the communication in TIL-Script is. The translation from natural language into TIL-Script messages (and vice versa) is near-to-isomorphic. Thus the humans can easily formulate the messages that the computational agents understand and perform.

Acknowledgements This research has been supported by the program "Information Society" of the Czech Academy of Sciences, project No. 1ET101940420 "Logic and Artificial Intelligence for Multi-Agent Systems"[2].

References

1. Duží M., Jespersen B., Materna P.: *Transparent Intensional Logic - Foundations and Applications*, mns. 2007.
2. Tichý P.: *The Foundations of Frege's Logic*, 1988, Berlin, New York: De Gruyter.
3. Tichý P.: *Cracking the Natural Language Code*, 1994 reprinted in [5]: pp. 843–857.
4. Tichý P.: *The Analysis of Natural Language*, 1994 reprinted in [5]: pp. 801–841.
5. *Pavel Tichý's Collected Papers in Logic and Philosophy* edited by V. Svoboda, B. Jespersen, C. Cheyne (eds.) Prague: Filosofia, Czech Academy of Sciences, and Dunedin: University of Otago Press.
6. Duží M.: *TIL as the Logic of Communication in a Multi-Agent System* Submitted to CICLING 2008.

[2] see `http://labis.vsb.cz`

Semantic Pre-processing of Anaphoric References

Marie Duží

VŠB-Technical University Ostrava, Czech Republic
marie.duzi@vsb.cz

Abstract. In the paper we describe the method of encoding communication of agents in a multi-agent system (MAS). The autonomous agents communicate with each other by exchanging messages formulated in a near-to-natural language. Transparent Intensional Logic (TIL) is an expressive system primarily designed for the logical analysis of natural language; thus we make use of TIL as a tool for encoding the semantic content of messages. The hyper-intensional features of TIL analysis are described in particular with respect to agents' attitudes and anaphoric references. By an example we illustrate the way TIL can function as a dynamic logic of discourse where anaphoric pronouns refer to entities of any type, even constructions, i.e. the structured meanings of other expressions.

1 Introduction

Multi-agent system (MAS) is a system composed of autonomous, intelligent but resource-bounded agents. The agents are active in their perceiving environment and acting in order to achieve their individual as well as collective goals. As a whole, the system of collaborative agents is able to deal with the situations that are hardly manageable by an individual agent or a monolithic centralised system. The agents communicate and collaborate with each other by exchanging messages formulated in a standardised natural language. According to the FIPA standards[1] for MAS, a *message* is the basic unit of communication. It can be of an arbitrary form but it is supposed to have a structure containing several attributes. Message semantic *content* is one of these attributes, the other being for instance 'Performatives', like 'Query', 'Inform', 'Request' or 'Reply'. The content can be encoded in any suitable language. The FIPA standard languages (for instance the SL language or KIF) are mostly based on the First-Order Logic (FOL) paradigm, enriched with higher-order constructs wherever needed. 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 focus on agents' communication, and we are going to demonstrate the need for an expressive logical tool of Transparent Intensional Logic (TIL) for encoding the semantic content of messages.

[1] The Foundation for Intelligent Physical Agents, http://www.fipa.org

Petr Sojka, Aleš Horák (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2007, pp. 43–56, 2007. © Masaryk University, Brno 2007

The paper is organised as follows. After briefly introducing TIL philosophy and motivations in the next Section 2, in Section 3 we describe the method of analysing sentences with anaphoric references occurring in any context; extensional, intensional, or even hyperintensional context of attitudes. By way of an example we demonstrate in Section 4 how TIL functions as the logic of dynamic discourse. Finally, a few notes on TIL implementation by the TIL-Script language are contained in concluding Section 5.

2 Basic notions of Transparent Intensional Logic

TIL *constructions* are uniquely assigned to expressions as their structured meanings. Intuitively, construction is a procedure (an instruction or a generalised algorithm), that consists of particular sub-instructions on how to proceed in order to obtain the output entity given some input entities. Thus the sense of a sentence is a hyper-proposition, i.e., the *construction* of a proposition denoted by the sentence. The denoted proposition is a flat mapping with the domain of possible worlds. Our motive for working 'top-down' has to do with anti-contextualism: any given term or expression (even one involving indexicals or anaphoric pronouns) expresses the same construction as its sense (meaning) in whatever sort of context the term or expression is embedded within. And the meaning of an expression determines the respective denoted entity (if any), but not vice versa. However, some terms, like those with indexicals or anaphoric pronouns, express only incomplete meanings (open constructions) and, therefore, only v(aluation)-denote, being insofar sensitive to which context they are embedded in.

There are two kinds of constructions, atomic and compound. Atomic constructions (*Variables* and *Trivializations*) do not contain any other constituent but itself; they supply objects (of any type) on which compound constructions operate. *Variables* $x, y, p, q, \ldots$, construct objects dependently on a valuation; they $v-$construct. *Trivialisation* of an object X(of any type, even a construction), in symbols $^0\boldsymbol{X}$, constructs simply Xwithout the mediation of any other construction. *Compound* constructions, which consist of other constituents, are *Composition* and *Closure*. *Composition* $[\boldsymbol{F}\ \boldsymbol{A}_1 \ldots \boldsymbol{A}_n]$ is the instruction to apply a function f ($v-$constructed by F) to an argument A ($v-$constructed by $A_1 \ldots A_n$).[2] Thus it $v-$constructs the value of f at A, if the function $f$is defined at A, otherwise the Composition is $v-$improper, i.e., it does not $v-$construct anything. *Closure* $[\lambda \boldsymbol{x}_1 \ldots \boldsymbol{x}_n\ \boldsymbol{X}]$ is the instruction to $v-$construct a function by abstracting over variables $x_1, \ldots, x_n$ in the ordinary manner of λ-calculi. Finally, higher-order constructions can be used twice over as constituents of composed constructions. This is achieved by a fifth construction called *Double Execution*, $^2\boldsymbol{X}$, that behaves as follows: If $X\, v-$constructs a construction X', and $X'\, v-$constructs an entity Y, then $^2X\, v-$constructs Y; otherwise ^2X is $v-$improper.

TIL constructions, as well as the entities they construct, all receive a type. The formal ontology of TIL is bi-dimensional; one dimension is made up of

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

constructions, the other dimension encompasses non-constructions. On the ground level of the type-hierarchy, there are entities unstructured from the algorithmic point of view belonging to a ***type of order 1***. Given a so-called *epistemic (or 'objectual')* ***base*** of ***atomic types*** (o-truth values, ι-individuals, τ-time moments / real numbers, ω-possible worlds), the induction rule for forming functions is applied: where α, β_1,…,β_n are types of order 1, the set of partial mappings from $\beta_1 \times \ldots \times \beta_n$ to α, denoted *(α $\beta_1 \ldots \beta_n$)*, is a type of order 1 as well.[3] Constructions that construct entities of order 1 are ***constructions of order 1***. They belong to a ***type of order 2***, denoted by $*_1$. This type $*_1$together with atomic types of order 1 serves as a base for the induction rule: any collection of partial mappings, type *(α $\beta_1 \ldots \beta_n$), involving* $*_1$ in their domain or range is a *type of order 2*. Constructions belonging to a type $*_2$ that identify entities of order 1 or 2, and partial mappings involving such constructions, belong to a ***type of order 3***. And so on *ad infinitum.*

An object A of a type α is called an α-object, denoted A/α. That a construction C $v-$constructs an α-object is denoted C $\rightarrow_v$ α. Quantifiers, $\forall^\alpha$ (the general one) and $\exists^\alpha$ (the existential one), are of types $(o(o\alpha))$, i.e., sets of sets of α-objects.[4] $[{}^0\forall^\alpha \lambda x A]$ $v-$constructs True iff $[\lambda x A]$ $v-$constructs the whole type α, otherwise False; $[{}^0\exists^\alpha \lambda x A]$ $v-$constructs True iff $[\lambda x A]$ $v-$constructs a non-empty subset of the type α, otherwise False. We write '$\forall x A$', '$\exists x A$' instead of '$[{}^0\forall^\alpha\ \lambda x A]$', '$[{}^0\exists^\alpha\ \lambda x A]$', respectively, when no confusion can arise. Singularisers ι^α are of types $(\alpha(o\alpha))$; $[{}^0\iota^\alpha\ \lambda x A]$ $v-$constructs the only α-member of the singleton $v-$constructed by $\lambda x A$, otherwise (i.e., if $\lambda x A$ v-constructs an empty class or a multi-element class) $v-$improper.

We use an infix notation without trivialisation when using constructions of truth-value functions $\wedge$ (conjunction), $\vee$ (disjunction), $\supset$ (implication), $\equiv$ (equivalence) and negation ($\neg$), and when using a construction of an identity.

(α-)intensions are members of a type $(\alpha\omega)$, i.e., functions from possible worlds to an arbitrary type α. *(α-)extensions* are members of the type α, where α is not equal to $(\beta\omega)$ for any β, i.e., extensions are not functions from possible worlds. Intensions are frequently functions of a type $((\alpha\tau)\omega)$, i.e., functions from possible worlds to *chronologies* of the type α (in symbols: $\alpha_{\tau\omega}$), where a chronology is a function of type $(\alpha\tau)$. We will use variables w, w_1, … as $v-$constructing elements of type ω (possible worlds), and t, t_1, … as $v-$constructing elements of type τ (times). If $C \rightarrow \alpha_{\tau\omega}$ $v-$constructs an α-intension, the frequently used Composition of the form $[[Cw]t]$, the intensional descent of the α-intension, is abbreviated as C_{wt}.

Some important kinds of intensions are:

Propositions, type $o_{\tau\omega}$. They are denoted by empirical (declarative) sentences.

[3] TIL is an open-ended system. The above epistemic base {o, ι, τ, ω} was chosen, because it is apt for natural-language analysis, but the choice of base depends on the area to be analysed. [4] Collections, sets, classes of 'α-objects' are members of type $(o\alpha)$; TIL handles classes (subsets of a type) as characteristic functions. Similarly relations (-in-extension) are of type(s) $(o\beta_1 \ldots \beta_m)$.

Properties of members of a type α, or simply *α-properties*, type $(o\alpha)_{\tau\omega}$. General terms (some substantives, intransitive verbs) denote properties, mostly of individuals.

Relations-in-intension, type $(o\beta_1 \ldots \beta_m)_{\tau\omega}$. For example transitive empirical verbs, also attitudinal verbs denote these relations.

α-roles, offices, type $\alpha_{\tau\omega}$, where $\alpha \neq (o\beta)$. Frequently $\iota_{\tau\omega}$. Often denoted by concatenation of a superlative and a noun (*"the highest mountain"*).

Example: We are going to analyse the sentence "Adam is looking for a parking place". Our method of analysis consists of three steps:

1. *Type-theoretical analysis*, i.e., assigning types to the objects talked about by the analysed sentence. In our case we have:
   (a) $Adam/\iota$;
   (b) $Look\_for/(o\iota(o\iota)_{\tau\omega})_{\tau\omega}$—the relation-in-intension of an individual to a property of individuals: the seeker wants to find an instance of the property;
   (c) $Parking(Place)/(o\iota)_{\tau\omega}$—the property of individuals.

2. *Synthesis*, i.e., composing the constructions of the objects *ad* (1) in order to construct the proposition of type $o_{\tau\omega}$ denoted by the whole sentence. The sentence claims that the individual Adam has the 'seeking-property' of looking for a parking place. Thus we have to construct the individual Adam, the 'seeking-property', and then apply the latter to the former. Here is how:
   (a) The atomic construction of the individual called Adam is simply ${}^0Adam$;

   (b) The 'seeking-property' has to be constructed by Composing the relation-in-intension *Look_for* with a seeker $x \rightarrow \iota$ and the property $Parking/(o\iota)_{\tau\omega}$ an instance of which is being sought: $[{}^0Look\_for_{wt} x \; {}^0Parking] \; v$—constructing a truth value. Abstracting first from x by $\lambda x[{}^0Look\_for_{wt} x \; {}^0Parking]$ we obtain the class of individuals; abstracting further from $w$and$t$ we obtain the 'seeking-property': $\lambda w \lambda t \; [\lambda x[{}^0Look\_for_{wt} x \; {}^0Parking]]$.
   (c) Now we have to Compose the property constructed *ad* (b) with the individual constructed *ad* (a). The property has to be subjected to the intensional descent first, i.e., $[\lambda w \lambda t \; [\lambda x[{}^0Look\_for_{wt} x \; {}^0Parking]]]_{wt}$ and then Composed with the former.[5] Since we are going to construct a proposition, i.e., an intension, we finally have to abstract from w, t:
   $$\lambda w \lambda t \; [[\lambda w \lambda t \; [\lambda x[{}^0Look\_for_{wt} x {}^0Parking]]]^0_{wt} Adam].$$
   This construction is the literal analysis of our sentence. It can be still β-reduced to the equivalent form:
   $$\lambda w \lambda t \; [{}^0Look\_for_{wt} \quad {}^0Adam \; {}^0Parking].$$

[5] For details on predication of properties and relations-in-intension of individuals, see Jespersen (forthcoming).

3. *Type-Theoretical checking*:

$$\begin{tikzcd}[row sep=small] \lambda w\lambda t \arrow[dd] & {[\ {}^0\mathit{Look\_for}_{wt}\ {}^0\mathit{Adam}\ {}^0\mathit{Parking}\ ]} \\ & \underbrace{(o\iota(o\iota)_{\tau\omega})\quad \iota\quad (o\iota)_{\tau\omega}}_{o} \arrow[dl] \\ o_{\tau\omega} & \end{tikzcd}$$

The role of Trivialisation and empirical parameters $w \to \omega$, $t \to \tau$ in the communication between agents can be elucidated as follows. Each agent has to be equipped with a basic ontology, namely the set of primitive concepts (Trivialised objects) she is informed about. Thus the upper index '^{0}' serves as a marker of the primitive concept that the agents should have in their ontology. If they do not, they have to learn them by asking the others. The lower index '$_{wt}$' can be understood as an instruction to execute an *empirical inquiry (search)* in order to obtain the actual current value of an intension, for instance by searching agent's database or by asking the other agents, or even by means of agent's sense perception.

3 Anaphora and Meaning

The problem of an anaphoric reference to a previously used expression is a well-known hard nut of *linguistic* analysis, because the antecedent of the anaphoric reference is often not unambiguously determined. Thus it is often said that anaphora constitutes a pragmatic problem rather than a problem of logical semantics. We agree that *logical* analysis cannot disambiguate any sentence, because it presupposes understanding and full linguistic competence. Yet our method of logical analysis can contribute to solving the problem of disambiguation in at least two respects; (a) a type-theoretical analysis often unambiguously determines which of the possible meanings of a homonymous expression is used in a sentence, and (b) if there are two or more possible readings of a sentence, the logical analysis should make all of them explicit. This often concerns the distinction between *de dicto* and *de re* readings.

In this section we propose a method of *logically* analysing sentences with anaphoric references. The method consists in substituting an appropriate construction of the object to which the anaphora refers for the anaphoric variable. In other words, we perform a *semantic* pre-processing of the embedded anaphoric clause based on the *meaning* of the respective antecedent. In this sense anaphora *is* a *semantic* problem.

3.1 Semantic pre-processing of Anaphoric References

Our hyperintensional (procedural) semantics makes it possible to apply anti-contextualist and compositional analysis to anaphoric sentences. The meaning of a sentence containing a clause with an anaphoric reference is the procedure which is a two-phase instruction that comes down to this:

(i) *execute the substitution based on the meaning of the antecedent for the anaphoric variable;*
(ii) *execute the result (a propositional construction) again to obtain a proposition.*

To specify phase (i) we make use of the fact that constructions are objects *sui generis* that the other constructions can operate on. The substitution is realised by a function $Sub/({}^{*}{}_{n}{}^{*}{}_{n}{}^{*}{}_{n}{}^{*}{}_{n})$ that operates on constructions C_1, C_2 and C_3 yielding as output the construction C_4 that is the result of substituting C_1 for C_2 in C_3. The phase (ii) consists in executing the adjusted meaning, namely the construction pre-processed by phase (i). To this end we use the fifth construction defined above, the *Double Execution.* The method is uniquely applicable to all kinds of sentences, including those that express (*de dicto* / *de re*) attitudes to a hyperintension, attitudes to an intension, and relations (-in-intension) to extensional entities. Now we adduce examples that illustrate the method. (A) "5 + 7 = 12, and Charles knows *it*."

The embedded clause "Charles knows it" does not express Charles' relation(-in-intension) to a truth-value, but to a *construction*, here the *procedure* of calculating the result of *5 + 7 = 12*. Hence $Know(ing)/(o\iota^{*}{}_1)_{\tau\omega}$ is a relation-in-intension of an individual to a construction.However, the meaning of the clause is incomplete; it is an *open* construction with the free variable *it*: $\lambda w\lambda t[{}^0Know_{wt}\ {}^0Charles\ it]$. The variable $it/{}^{*}{}_2 \rightarrow *_1$ is the meaning of the pronoun '*it*' that in (A) anaphorically refers to the meaning of "5 + 7 = 12", i.e., the construction $[{}^0{+}\,{}^05\ {}^07]$. The meaning of the whole sentence (A) is, however, complete. It is the *closed* construction

(A′) $\lambda w\lambda t[[{}^0 = {}^0{+}\,{}^05\,{}^07]\ {}^012] \wedge$
$\qquad {}^{z}[{}^0Sub\ {}^{00}[{}^0 = {}^0{+}\,{}^05\,{}^07]\ {}^012]\ {}^0it\ {}^0[\lambda w\lambda t[{}^0Know_{wt}\ {}^0Charles\ it]]]_{wt}]$

Types: $Charles/\iota$; $Know/(o\iota^{*}{}_1)_{\tau\omega}$; $Sub/({}^{*}{}_2{}^{*}{}_2{}^{*}{}_2{}^{*}{}_2)$; $it/{}^{*}{}_2 \rightarrow {}^{*}{}_1$; the other types are obvious.

Since (A′) seems to be rather complicated, we now show that (A′) is an adequate analysis meeting our three requirements of compositionality, anti-contextualism and a purely semantic solution. The argument of the second conjunct of (A′), namely

(S) $[{}^0Sub\ {}^{00}[{}^0 = {}^0{+}\,{}^05\,{}^07]\ {}^012]\ {}^0it\ {}^0[\lambda w\lambda t[{}^0Know_{wt}\ {}^0Charles\ it]]]_{wt}] \rightarrow {}^{*}{}_1$

constructs a *construction* of order 1, namely the one obtained by the substitution of the construction ${}^0[{}^0 = {}^0{+}\,{}^05\,{}^07]\ {}^012]$ for the variable *it* into the construction $[\lambda w\lambda t[{}^0Know_{wt}\ {}^0Charles\ it]]$.The result is the construction

(S′) $[\lambda w\lambda t[{}^0Know_{wt}\ {}^0Charles\ {}^0[{}^0 = {}^0{+}\,{}^05\,{}^07]\ {}^012]]]$,

which constructs a proposition P. But an argument of the truth-value function conjunction ($\wedge$) can be neither a propositional construction, nor a proposition, but must be a truth-value. Since (S) constructs the construction (S′), and (S′) constructs the proposition P, the execution steps have to be: (a) execute (S) to obtain the propositional construction (S′), (b) execute the result (S′) to obtain the proposition P; hence we need the Double Execution of (S) to construct the proposition P, and then (c) P has to undergo intensional descent with respect to the external w, t in order to $v-$construct a truth-value.

Note that the open construction $\lambda w \lambda t\ [{}^0 Know^0_{wt} Charles\ it]$ is assigned to "Charles knows *it*" invariably of a context. The variable *it* is free here either for a pragmatic valuation or for a substitution by means of the meaning of the antecedent that is referred to in the linguistic context. The object—*what* is known by Charles—can be completed by a situation of utterance or by a linguistic context. If the sentence occurs within another linguistic context, then *Sub* substitutes a different construction for the variable *it*, namely the construction to which '*it*' anaphorically refers.

The other example concerns Charles' attitude of seeking the occupant of an individual office:

(B) "Charles sought the Mayor of Dunedin but (*he*) did not find *him*."

Suppose now the *de dicto* reading of (B), i.e., that Charles' search concerned the office of Mayor of Dunedin and not the location of its holder. The function *Sub* creates a new construction from constructions and, so, can easily be iterated. The analysis of (B) is:

(B^d) $\lambda w \lambda t\ [[{}^0 Seek_{wt}\ {}^0 Ch\ \lambda w \lambda t\ [{}^0 Mayor\_of_{wt}\ {}^0 D]] \wedge {}^2[\ {}^0 Sub\ {}^{00} Ch\ {}^0 he$
$[{}^0 Sub\ {}^0[\lambda w \lambda t\ [{}^0 Mayor\_of_{wt}\ {}^0 D]]\ {}^0 him\ {}^0[\lambda w \lambda t \neg [{}^0 Find_{wt}\ he\ him]]]]_{wt}]$.

Types: $Seek/(o\iota_{\tau\omega})_{\tau\omega}$; $Find/(o\iota_{\tau\omega})_{\tau\omega}$; $Ch(arles)/\iota$; $Mayor\_of$ (something)/ $(\iota\iota)_{\tau\omega}$; D(unedin)$/\iota$; $he/*_1 \rightarrow \iota$; $him/*_1 \rightarrow \iota_{\tau\omega}$.

Again, the meaning of (B) is the closed construction (B^d), and the meaning of the embedded clause "*he* did not find *him*" is the open construction[6] $\lambda w \lambda t \neg [{}^0 Find_{wt}\ he\ him]$ with the two free variables *he* and *him*.

Of course, another refinement is thinkable. The variables *he* and *him*, ranging over individuals and individual offices, respectively, reduce the ambiguity of 'find' by determining that here we are dealing with finding the occupant of an individual office. But the expressions like 'he', 'him', or 'she', 'her' also indicate that the finder as well as the occupant of the sought office are male and female, respectively. Thus, e.g., a refined meaning of "He found her" might be

$\lambda w \lambda t\ [[{}^0 Find_{wt}\ he\ her] \wedge [{}^0 Male_{wt} he] \wedge [{}^0 Female_{wt}\ her_{wt}]]$.

Additional types: $Male, Female/(o\iota)_{\tau\omega}$; $her/*_1 \rightarrow \iota_{\tau\omega}$.

Now perhaps a more natural *de re* reading (B^r) of the sentence (B) is understood as uttered in a situation where Charles knows who the Mayor is, and is striving to locate this individual. Unlike the *de dicto* case, the sentence understood *de re* has an *existential presupposition*: in order that (B^r) have *any* truth value, the Mayor has to exist. Thus we must not substitute the construction of an office, but of the individual (if any) that occupies the office. To this end we use $[{}^0 Tr\ [{}^0 Mayor\_of^0_{wt} D]]$ that fails to construct anything if $[{}^0 Mayor\_of^0_{wt} D]$ is $v-$improper (the Mayor does not exist), otherwise it $v-$constructs the Trivialisation of the occupant of the office. Using the technique of substitutions we can discover the adequate analysis of (B^r). Here is how:

[6] Tenses are disregarded.

$\lambda w\lambda t[[{}^0Seek^L{}_{wt}{}^0Charles\ {}^2[{}^0Sub\ [{}^0Tr\ [{}^0Mayor\_of_{wt}{}^0D]]\ {}^0who\ {}^0[\lambda w\lambda t\ [{}^0Loc_{wt}\ who]]]] \wedge {}^2[{}^0Sub^{00}Charles\ {}^0he\ [{}^0Sub\ [{}^0Sub\ [{}^0Tr\ [{}^0Mayor\_of_{wt}{}^0D]]\ {}^0who\ {}^0[\lambda w\lambda t\ [{}^0Loc_{wt}who]]]\ {}^0it\ {}^0[\lambda w\lambda t\neg[{}^0Find^L{}_{wt}he\ it]]]]_{wt}]$

Types: $Seek^L, Find^L/(o\iota\mu_{\tau\omega})_{\tau\omega}; Tr/(*_1\iota); Charles/\iota; Mayor\_of$ (something)$/(\iota\iota)_{\tau\omega}$; D(unedin)$/\iota$; $he, who/*_1 \rightarrow \iota; it/*_1 \rightarrow \mu_{\tau\omega}; Loc/(\mu\iota)_{\tau\omega}$.[7]

The second conjunct, which is rather more complicated, needs a gloss. Here we have to pre-process by substitution the meaning of the second embedded clause "he did not find it", i.e. the open construction $[\lambda w\lambda t\neg[{}^0Find^L_{wt}he\ it]]$, by substituting the construction that has been sought, i.e., the location of the individual who plays the role of Mayor of Dunedin: $[{}^0Sub\ [{}^0Tr\ [{}^0Mayor\_of^0_{wt}D]]\ {}^0who\ {}^0[\lambda w\lambda t\ [{}^0Loc_{wt}who]]]$.

3.2 Donkey Sentences

The following example is a variant of the well-known problem of Peter Geach's *donkey sentences*:

(D) "If somebody has got a new car then *he* often washes *it*."

The analysis of the embedded clause "*he* often washes *it*" containing the anaphoric pronouns 'he' and 'it' is again an open construction with two free variables *he*—*who* (washes), *it*—*what* (is washed), $he, it \rightarrow \iota$; $Wash/(o\iota\iota)_{\tau\omega}$:

$\lambda w\lambda t[{}^0Wash_{wt}\ he\ it]$.

If we also want to analyze the frequency of washing, i.e., the meaning of 'often', then we use the function $Freq(uently)/((o(o\tau))\tau)$. The function *Freq* associates each time T with a set of those time intervals (of type $(o(o\tau))$) that are frequent in T (for instance, once a week). The analysis of "*he* often washes *it*" is then

$\lambda w\lambda t\ [{}^0Freq_t\ \lambda t'[{}^0Wash_{wt'}\ he\ it]]$.

However, since rendering the frequency of washing does not influence the way of solving the problem of anaphora in donkey sentences, we will use, for the sake of simplicity, the simpler construction $\lambda w\lambda t[{}^0Wash_{wt}\ he\ it]$.

The problem of donkey sentences consists in discovering their logical form, because it is not clear how to understand them. Geach (1962, p. 126) proposes a structure that can be rendered in 1st-order predicate logic as follows (*NC* new car):

$\forall x\forall y((NC(y) \wedge Has(x, y)) \rightarrow Wash(x, y))$.

However, Russell objected to this analysis that the expression 'a new car' is an *indefinite description*, which is not rendered by Geach's analysis. Hence Russell proposed an analysis that corresponds to this formula of 1st-order predicate logic:

$\forall x\ (\exists y\ (NC(y) \wedge Has(x, y)) \rightarrow Wash(x, \boldsymbol{y}))$.

[7] The type μ is the type of a location/position.

But the last occurrence of the variable y (marked in bold) is free in this formula—out of the scope of the existential quantifier supposed to bind it.

Neale in his (1990) proposes a solution that combines both of the above proposals. On the one hand, the existential character of an indefinite description is saved (Russell's demand), and on the other hand, the anaphoric variable is bound by a general quantifier (Geach's solution). Neale introduces so-called *restricted quantifiers*:[8]

[every x: man x and [a y: new-car y](x owns y)]([whe z: car z and x owns z] (x often washes z)).

The sentence (D) does not entail that if the man owns more than one new car then some of this cars are not washed by him. Hence we can reformulate the sentence into

(D_1) "Anybody who owns some new cars often washes *all of them* [each of the new cars he owns]."

However, the following sentence (D") means something else:

(D_2) "Anybody who owns some new cars often washes *some of them* [some of the new cars he owns]."

The analysis of (D_1), which in principle corresponds to Geach's proposal, is

(D_1') $\lambda w \lambda t \forall x \forall y [[[{}^0NC_{wt} y] \wedge [{}^0Own_{wt}\ x\ y]] \supset$
$${}^2[{}^0Sub\ {}^0x {}^0he\ [{}^0Sub\ {}^0y {}^0it\ {}^0[\lambda w \lambda t [{}^0\mathrm{Wash}_{wt}\ he\ it]]]]_{wt}].$$

Types: $Own/(o\iota\iota)_{\tau\omega}$; $Wash/(o\iota\iota)_{\tau\omega}$; NC (being a new car)$/(o\iota)_{\tau\omega}$; $x, y, he, it \rightarrow \iota$.

But then an objection due to Neale can be levelled against these analyses, namely that in the original sentence (D) the anaphoric pronoun 'it' stands *outside* of the scope of the quantifier occurring in the antecedent. To overcome this objection, we use a different type of quantifiers. Apart the common quantifiers $\forall, \exists\ /\ (o(o\iota))$ that operate on a set of individuals, we use quantifiers of another type, namely *Some* and $All/((o(o\iota))(o\iota))$. *Some* is a function that associates the argument—a set S—with the set of all those sets which have a non-empty intersection with S. *All* is a function that associates the argument—a set S—with the set of all those sets which contain S as a subset. Thus for instance the sentence "Some students are happy" is analyzed by

$\lambda w \lambda t\ [[{}^0Some\ {}^0Student_{wt}]\ {}^0Happy_{wt}]$.

The analyses of the embedded clauses of (D_1), (D_2), namely "*he* washes all of *them*", "*he* washes some of *them*" are (the anaphoric pronoun '*them*' refers here to the *set of individuals*; we use the variable $them \rightarrow (o\iota)$ as the meaning of 'them')

$\lambda w \lambda t\ [[{}^0All\ them]\ \lambda it[{}^0Wash_{wt}\ he\ it]]$, $\lambda w \lambda t\ [[{}^0Some\ them]\ \lambda it[{}^0Wash_{wt}\ he\ it]]$

respectively. Now we need to substitute a construction of the set of new cars owned by the man for the variable *them.* Further, we have to substitute the

[8] Neale (1990, p. 236). Neale takes into account that the sentence is true even if a man owns *more than one* new car. To avoid singularity he thus claims that the description used in his analysis does not have to be singular (definite) but plural: his abbreviation 'whe *F*' stands for 'the *F* or the *F*s'.

variable x ('anybody') for the variable *he* ('*who* washes'), and then the pre-processed construction has to be Double Executed. To prevent collision of variables, we rename the internal variables w, t.

(D$_1$″) $\lambda w\lambda t\ [^0\forall\lambda x\ [[[^0Man_{wt}x] \wedge [^0\exists\lambda y[[^0NC_{wt}y] \wedge [^0Own_{wt}\ x\ y]]]] \supset$
$^2[^0Sub\ ^0[\lambda y[[^0NC_{wt}y] \wedge [^0Own_{wt}\ x\ y]]]\ ^0them\ [^0Sub^0x^0he$
$^0[\lambda w'\lambda t'\ [[^0All\ them]\ \lambda it[^0Wash_{w't'}\ he\ it]]]]]_{wt}]].$

Gloss: "For every man, if the man owns some new cars then all of them [i.e., the new cars owned] are washed by him [the man x]."

This construction can be viewed as the most adequate analysis of (D$_1$), because it meets Russell's requirement of an indefinite description in the antecedent, while the scope of $\exists$ does not exceed the antecedent.

The second possible reading of (D) is now analyzed using *Some* instead of *All*:

(D$_2$″) $\lambda w\lambda t\ [^0\forall\lambda x\ [[[^0Man_{wt}x] \wedge [^0\exists\lambda y[[^0NC_{wt}y] \wedge [^0Own_{wt}\ x\ y]]]] \supset$
$^2[^0Sub\ ^0[\lambda y[[^0NC_{wt}y] \wedge [^0Own_{wt}\ x\ y]]]\ ^0them\ [^0Sub^0x^0he$
$^0[\lambda w'\lambda t'\ [[^0Some\ them]\ \lambda it[^0Wash_{w't'}\ he\ it]]]]]_{wt}]].$

Gloss: "For every man, if the man owns some new cars then some of them [i.e., the new cars owned] are washed by him [the man x]."

As we pointed out above, it is not clear how to exactly understand the sentence (D), simply because the sentence is ambiguous. We thus offered analyses that disambiguate it. Whether these readings are the only possible ones is not for us to decide. In our opinion the reading (D$_1$) is more plausible, and Neale takes into account only this one. However, our method makes it possible to easily analyse particular variants of donkey sentences like "… most of them…", and suchlike. It might be objected, however, that in the interest of disambiguation, we actually analysed two variants of the original sentence.

Sandu formulates in (1997) two principles that every 'compositional procedure for analysing natural language sentences' should obey:

(a) there is a one-to-one mapping of the surface structure of a sentence of (a fragment of) English into its logical form which preserves the left-to-right ordering of the logical constants
(b) the mapping preserves the nature of the lexical properties of the logical constants, in the sense that an indefinite is translated by an existential quantifier, etc.

One can see that our analyses (D$_1$″) and (D$_2$″) obey these principles with respect to the glossed variants, but not with respect to the original sentence (D). Regardless of the disambiguation concerning some/all new cars being washed, principle (b) is violated because 'a man' is analysed as 'every man'. To put our arguments on a still more solid ground, we now propose the literal analysis of the sentence (D). The analysis of the clause "A man has a new car" is as follows:

(NC) $\lambda w\lambda t\ [^0\exists\lambda xy\ [[^0Man_{wt}x] \wedge [^0NC_{wt}y] \wedge [^0Own_{wt}\ x\ y]]].$

Additional type: $\exists/(o(o\iota\iota))$.

The consequent of (D) expresses that *all* the couples <*he*, *it*> are such that *he Washes it.* Using a variable *couples*$/*_1 \rightarrow (ο\iota)$, we have:

$\lambda w \lambda t[[^0 \mathit{Allcouples}]\ \lambda \mathit{he}\ \mathit{it}\ [^0 \mathit{Wash}_{wt}\ \mathit{he}\ \mathit{it}]]$.

Now composing (NC) with the latter, we substitute the construction of the set of couples constructed by the Closure of (NC) for the variable *couples*:

(D′) $\lambda w \lambda t[[^0 \exists \lambda xy\ [[^0 \mathit{Man}_{wt} x] \wedge [^0 \mathit{NC}_{wt} y] \wedge [^0 \mathit{Own}_{wt}\ x\ y]]] \supset$
$^2[^0 \mathit{Sub}\ ^0[\lambda xy\ [[^0 \mathit{Man}_{wt} x] \wedge [^0 \mathit{NC}_{wt} y] \wedge [^0 \mathit{Own}_{wt}\ x\ y]]]\ ^0 \mathit{couples}$
$^0[\lambda w \lambda t[[^0 \mathit{All\ couples}]\ \lambda \mathit{he}\ \mathit{it}\ [^0 \mathit{Wash}_{wt}\ \mathit{he}\ \mathit{it}]]]]_{wt}]$.

As is seen, (D′) is fully compositional. Our constituents operate on constructions of sets of couples of individuals, as well as particular individuals, which is impossible within a first-order theory. In this respect Hintikka is right when claiming that the compositional treatment does not work;[9] it does not work within a first-order framework. But as soon as we have a powerful higher-order system like TIL at our disposal, there is no need to give up the desirable principle of compositionality.

One pressing question is whether the anaphoric pronouns should be, in general, bound, and if so, another pressing question is whether this is to be in a standard or non-standard way. The Dynamic Predicate Logic (DPL) applies a mechanism of passing on binding.[10] Note that (D′) at the same time provides the semantics of this mechanism. Indeed, the variables *he* and *it* are bound in (D′), but the binding is of another kind. They are not directly bound by the existential quantifier. Technically, they are bound by Trivialization; semantically, they are bound by the condition that the pairs of individuals they $v-$construct have to belong to the set mentioned by the antecedent clause.

4 Outline of an Implementation Method

Now we outline the method of computing the complete meaning of anaphoric sentences, i.e., the method of substituting an appropriate antecedent for an anaphoric reference. The method is similar to the one applied in general by Hans Kamp's Discourse Representation Theory (DRT). 'DRT' is an umbrella term for a collection of logical and computational linguistic methods developed for dynamic interpretation of natural language, where each sentence is interpreted within a certain discourse, which is a sequence of sentences uttered by the same speaker. Interpretation conditions are given *via* instructions for updating the discourse representation. DPL is a logic belonging to this group of theories. Discourse representation theory as presented in Kamp & Reyle (1993) addresses in particular the problem of anaphoric links crossing the sentence boundary. It is a first-order theory, and it can be proved that the expressive power of the DRT language with negation is the same as that of first-order predicate logic. Thus actually only expressions denoting individuals (indefinite or definite noun phrases) introduce the so-called discourse referents, i.e., free variables that are updated when interpreting the discourse. Anaphoric pronouns

9 See Sandu & Hintikka (2001) 10 See Sandu (1997).

are represented by free variables linked to appropriate antecedent discourse variables. As we have seen above, our semantics is hyperintensional, i.e., procedural, and higher order. Thus not only individuals, but entities of any type, like properties of individuals, propositions, relations-in-intension of an individual to another individual, and even constructions (i.e. meanings of the antecedent expressions), can be linked to anaphoric variables.

The specification of the implementation algorithm proposed here is imperative; similarly as in DRT, we update the list of potential antecedents, or rather constructions expressed by them, in order to substitute the type-appropriate entities for anaphoric variables, whenever needed.[11] For each type (ι, $(o\iota)_{\tau\omega}$, $o_{\tau\omega}$, $(o\iota(o\iota)_{\tau\omega})_{\tau\omega}$, $(o\iota\iota)_{\tau\omega}$, $*_n$, etc.) the list of discourse variables is created. The method substitutes the content of type-appropriate discourse variables for anaphoric variables to complete the meaning of anaphoric clauses. Each closed constituent of a resulting construction becomes an updated value of the respective (type-appropriate) free discourse-referent variable. In this way the discourse variables are gradually updated.

Here we only illustrate the method by an example of a simple dialog between three agents, *Adam*, *Berta* and *Cecil*. The list of discourse variables for the dialog together with the types of entities constructed by their respective content is: $ind{:=}\iota$, $loc{:=}\mu$, $pred{:=}(o\iota)_{\tau\omega}$, $prof{:=}(o\iota)_{\tau\omega}$—'propositional function', $rel_1{:=}(o\iota(o\iota)_{\tau\omega})_{\tau\omega}$, $rel_2{:=}(o\iota\iota)_{\tau\omega}$, $rel_3{:=}(o\iota o_{\tau\omega})_{\tau\omega}$, $prop{:=}o_{\tau\omega}$, $constr{:=}*_n$.

Adam to Cecil: "Berta is coming. ***She*** is looking for a parking".
'Inform' message content:
$\lambda w\lambda t[[{}^0Coming^0_{wt}Berta];$
(Relevant) discourse variables updates:
$ind{:=}{}^0Berta;\ pred{:=}{}^0Coming;$
$prop{:=}\ \lambda w\lambda t[[{}^0Coming^0_{wt}Berta];$
$\lambda w\lambda t\ \ {}^2[{}^0Sub\ ind\ {}^0she\ {}^0[{}^0Looking\_for_{wt}she\ {}^0Parking]] \Rightarrow$ (is transformed into)
$\lambda w\lambda t[{}^0Looking\_for^0_{wt}Berta\ {}^0Parking].$
(Relevant) discourse variables updates:
$rel_1{:=}{}^0Looking\_for;\ pred{:=}{}^0Parking;$
$prop{:=}\ \lambda w\lambda t[{}^0Looking\_for^0_{wt}Berta\ {}^0Parking];$
$prof{:=}\ \lambda w\lambda t\lambda x[{}^0Looking\_for_{wt}x{}^0Parking];$ ('propositional function')

Cecil to Adam: "***So*** am I."
'Inform' message content:
$\lambda w\lambda t^2[{}^0Sub\ prof\ {}^0so\ {}^0[so^0_{wt}Cecil]] \Rightarrow \quad \lambda w\lambda t[{}^0Looking\_for^0_{wt}Cecil\ {}^0Parking]$
Discourse variables updates:
$ind{:=}{}^0Cecil;\ rel_1{:=}{}^0Looking\_for;\ pred{:=}{}^0Parking;$
Adam to both: "There is a free parking at p_1".
'Inform' message content: $\lambda w\lambda t[[{}^0Free\ {}^0Parking]_{wt}\ {}^0p_1]$
Discourse variables updates: $loc{:=}{}^0p_1;\ pred{:=}[{}^0Free\ {}^0Parking];$
$prop{:=}\ \lambda w\lambda t[[{}^0Free\ {}^0Parking]_{wt}\ {}^0p_1]$
Berta to Adam: "What do you mean by free parking?"
'Query' message content: $\lambda w\lambda t\ [{}^0Refine_{wt}\ \ {}^0[{}^0Free\ {}^0Parking]]$

[11] The algorithm was first proposed in Křetínský (2007).

Discourse variables updates: *constr*:=$^0[^0Free\ ^0Parking]$

Adam to Berta: "Free parking is a parking and some parts of it are not occupied".
'Reply' message content: $^0[^0Free\ ^0Parking] =$
$^0[\lambda w\lambda t\lambda x[[^0Parking_{wt}x] \wedge \ \exists y[[^0Part\_of_{wt}yx] \wedge \ \neg[^0Occupied_{wt}y]]]]$
Discourse variables updates: *constr*:=$^0[^0Free\ ^0Parking] = \ldots$

Berta to Adam: "I don't believe ***it***. I have just been ***there***".
'Inform' message content:
$\lambda w\lambda t\ [^2[^0Sub\ prop\ ^0it\ ^0[\neg[^0Believe_{wt}\ \ ^0Berta\ it]]] \Rightarrow$
$\lambda w\lambda t\ \ \neg[^0Believe_{wt}\ ^0Berta\ ^0[\lambda w\lambda t[[^0Free\ ^0Parking]_{wt}\ p_1]]]$,
Discourse variables updates:
ind:=^0Berta; *loc*:=$^0p_1$;
$\lambda w\lambda t\ \ \exists t'[[t' \leq\ \ t] \wedge {}^2[^0Sub\ loc\ ^0there\ ^0[^0Is\_at^0_{wt'}Berta\ there]]] \Rightarrow$
$\lambda w\lambda t\exists t'[[t' \leq\ \ t] \wedge [^0Is\_at^0_{wt'}Berta\ ^0p_1]]$.
Discourse variables updates:
prop:= $\lambda w\lambda t\exists t'[[t' \leq\ \ t] \wedge [^0Is\_at^0_{wt'}Berta\ ^0p_1]], \ldots$
And so on.

Of course, improvements of this method are straightforward. For instance, in the example we were substituting the last type-appropriate entity that received mention; if we wanted to take into account ambiguities of anaphoric references, we might store in the discourse-representation file more than one variable for each type, together with the other characteristics or prerequisites of the entity (e.g., gender, or implicative properties), so as to be able to generate more meanings of an ambiguous sentence.

5 Concluding Remarks

The above described method is currently being implemented in the TIL-Script programming language, the computational variant of TIL. TIL-Script is a FIPA compliant higher-order modification of the standards like FIPA SL (Semantic Language) and FIPA KIF (Knowledge Interchange Format). It is a declarative functional language. Its only imperative feature is the *Let* command for the dynamic assignment of a construction *C* to a discourse variable. A brief introduction to TIL-Script is the subject of another paper in this proceedings, namely 'TIL-Script: Functional Programming Based on Transparent Intensional Logic' by Nikola Ciprich, Marie Duží, and Michal Košinár.

Acknowledgements

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

References

1. Geach, P.: *Reference and Generality.* Ithaca, NY: Cornell University Press (1962).
2. Kamp, H. and Reyle, U.: *From Discourse to Logic. Introduction to Model-Theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory.* Springer (1993).
3. Jespersen, B.: 'Predication and extensionalization'. *Journal of Philosophical Logic*, forthcoming (accepted 24 October 2007).
4. Křetínský, J. (2007): *Use-mention Distinction in Transparent Intensional Logic*. Bachelor thesis, Masaryk University Brno. Retrievable at: `http://is.muni.cz/th/139914/fi_b/bachelor.pdf`
5. Materna, P.: *Conceptual Systems.* Logos Verlag, Berlin (2004).
6. Neale, S.: *Descriptions*. The MIT Press, Cambridge (1990).
7. Rescher, N.: *Epistemic Logic.* Pittsburgh: University of Pittsburgh Press (2005).
8. Sandu, G.: 'On the theory of anaphora: dynamic predicate logic vs. game-theoretical semantics'. *Linguistic and Philosophy* 20 (1997), 147–174.
9. Sandu, G., Hintikka, J.: 'Aspects of compositionality'. *Journal of Logic, Language, Information 10* (2001) 49–61.
10. Tichý, P.: *The Foundations of Frege's Logic*, Berlin, New York: De Gruyter (1988).
11. Tichý, P.: *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 (2004).

Enhancing Anaphora Resolution for Czech

Vašek Němčík

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

Abstract. Resolution of anaphoric reference is one of the most important challenges in natural language processing (NLP). Functionality of most NLP systems crucially relies on an accurate mechanism for determining which expressions in the input refer to the same entity in the real world. The immense complexity of this issue has led the research community to adopt predominantly knowledge-poor methods, despite the fact that these are known to be incapable of solving this task reliably. This paper suggests several ways of extending such methods by further resources and mechanisms in order to arrive at a more adequate anaphora resolution procedure.

1 Introduction

Anaphora has been one of the most intensively studied issues in the linguistic research over the past decades. It has been studied from many different perspectives – from the point of view of syntax, semantics, pragmatics, psycholinguistics, computational linguistics, rhetoric, logic, philosophy, etc. Nevertheless, we still seem to be left in the dark about many important aspects of anaphora.

This situation is well apparent from the recent implementations of anaphora resolution (henceforth AR) systems. Since the mid-1990s, most of the implementations have been based on knowledge-poor and machine learning (ML) approaches, relying solely on low-level features such as morphological tags and shallow syntactic labels.[1] This trend is motivated by practical reasons. The individual low-level features can be computed automatically, efficiently and with sufficient accuracy. In contrast, higher-level information, such as a full syntactic parse or underlying semantics, unfortunately can't be obtained reliably enough, and the consequent errors undermine the AR performance considerably.

Knowledge-poor systems have proven themselves as a sensible trade-off between accuracy and computational feasibility. On the other hand, higher-level information is known to play an important role in anaphoric relations and thus can't be ignored altogether. This can be illustrated by the following

[1] The most influential knowledge-poor systems are systems presented by Lappin and Leass ([7]), Kennedy and Boguraev ([5]), Baldwin ([1]), and Mitkov et al. ([12]). ML-based systems will be discussed in section 3.

examples that demonstrate the necessity of semantic information and world knowledge for proper treatment of anaphora:

(1) a. After the bartender$_i$ served the patron$_j$, he$_i$ got a big tip.
b. After the bartender$_i$ served the patron$_j$, he$_j$ left a big tip.

(2) If the baby$_i$ does not thrive on raw milk$_j$, boil it$_{*i,j}$.

Although it is obvious that obtaining and combining all types of information relevant to AR is well beyond the scope of today's science, it is worthwhile to use at least certain types of higher-level information. This paper proposes how this can be done for Czech.

Next section suggests how to take advantage of certain more sophisticated linguistic resources to improve the performance of AR. Further, section 3 suggests several ways of adapting AR methods based on machine learning so that they grasp the properties of anaphoric relations in a more plausible way.

2 Exploiting Linguistic Resources for AR

This section gives a number of hints how to extend the common knowledge-poor systems by considering various kinds of higher-level information. Of course, the possibilities depend on what resources are available for the language in question. As this article concerns anaphora resolution with regard to Czech, it reflects particular resources available for Czech. Nonetheless, the following ideas can be straightforwardly applied to any language for which similar resources exist.

To my knowledge, at the moment, there are three AR systems for Czech. The first one is the modular system proposed by Němčík ([13]), encompassing selected salience-based algorithms. The other two systems were presented by Linh ([9]) – one of them is rule-based and the other is based on machine learning. All of these systems take advantage of solely knowledge-poor features and can be straightforwardly extended to use further resources.

The desired extension would ideally help to rule out semantically implausible antecedent candidates that would get otherwise incorrectly chosen by the original system.[2] Not necessarily all such antecedents need to be ruled out, on the other hand, it is important that the enhancing mechanism in question be sound, i.e. it shouldn't rule out correct antecedents.

The first potentially useful resource available for Czech is the Czech Wordnet, described by Pala and Smrž ([15]).[3] Its English version is often used to determine semantic plausibility when dealing with coreference resolution. However, on its own, it is rather useless for resolving pronominal anaphora.

[2] This idea has already been mentioned by Hobbs ([4]). [3] Strube and Ponzetto ([17]) argue that for practical purposes Wikipedia is a more useful resource, because it doesn't suffer from problems of hand-crafted taxonomies and contains information not only about classes but also idividual real-world instances. Moreover, it is larger and grows faster than Wordnet.

Another type of resource that could be used within the AR process are valency lexicons. For Czech, two of them are available, Vallex and Verbalex.[4] In my opinion, especially Verbalex is very helpful in this cause because its valency slots are annotated with semantic constraints. These are marked using Wordnet synsets, meaning that each slot can be filled only by an concept that is a hyponym of the synset indicated. This can be straightforwadly used in combination with Wordnet as a semantic plausibility check for AR illustrated by the following schema:

(3) a. $Verb_1\ \alpha_1\ \ldots\ \alpha_{i-1}\ Y\ \alpha_{i+1}\ \ldots$
. . .
b. $Verb_2\ \beta_1\ \ldots\ \beta_{j-1}\ X\ \beta_{j+1}\ \ldots$

Let us assume that X is an anaphor and Y an antecedent candidate preceding it.[5] Should Y be a plausible antecedent for X, it should meet the restrictions posed on the valency slot of X. In particular, it should be a hyponym of the synset associated with this valency slot. This mechanism can contribute to the correct resolution of anaphors in the following examples:[6]

(4) a. ***Obsluhující robot***$_i$ odnesl prázdnou misku od ***ovoce***$_j$
Robot (MASC.SG.) took the empty bowl of fruits (NEUT.SG.)
"The robot took away the empty fruit bowl"
b. a Alvar si teprve díky tomu uvědomil,
and Alvar only then thanks to this realized,
"and only after noticing this Alvar realized"
c. že ***ho***$_{*i,j}$ vůbec snědl.
that him/it (MASC./NEUT.SG.) actually ate.
"that he actually ate it."

(5) a. Dolehl k ***němu***$_i$ ***zvuk***$_j$
echoed to him (MASC.SG.) sound (MASC.SG.)
melodického smíchu
of melodic laughter
"A sound of melodic laughter echoed to him"
b. a $\emptyset_{i,*j,*k}$ na okamžik si myslel,
and [he (MASC.SG.)] for a moment thought,
"and for a moment he thought"
c. že je to Mary.
that is it Mary.
"it was Mary."

[4] For more information about Vallex and Verbalex please refer to Lopatková et al. ([10]), and Hlaváčková and Horák ([3]), respectively. [5] It may well be that $Verb_1 = Verb_2$, that is, that the anaphor and the antecedent are in the same clause. [6] The examples are for the sake of brevity slightly abridged sentences taken from The Czech National Corpus ([18]).

Obviously, this mechanism is not applicable to all anaphor–antecedent candidate pairs of this kind. The potential hindrances are many – it is not possible to reliably assign a unique valency frame to every sentence, to disambiguate every relevant word and match it with the correct Wordnet synset, and most importantly, neither Wordnet nor Verbalex can cover all words. However, to obtain a similar effect with higher recall, we can engage methods for determining semantic relatedness.

Recently, many interesting corpus-based methods have been proposed that make it possible to measure semantic similarity between words. For instance, Lin ([8]) has formulated a similarity measure based on mutual information between words.[7] A similar measure is adopted in the Sketch Engine tool (Kilgarriff et al., [6]) and can be utilized to approximate suitability of verb–argument combinations. This allows making a more sophisticated choice among top antecedent candidates. As a result, many resolution errors can be avoided, especially in cases when there is only a small difference in salience among top antecedent candidates.

The above-mentioned mechanisms seem to be a very promising first step in integrating semantics into AR systems. Investigation of their potential in practice is subject of my future work.

3 Anaphora Resolution and Machine Learning

This section suggests how AR approaches based on ML can be altered to more closely reflect the properties of anaphoric reference.

Presently, methods based on ML form an integral part of the mainstream AR research. Nevertheless, ML methods are not directly applicable to the AR task, because its structure is unsuitable and it needs to be transformed first to fit the ML concept. To my knowledge, two notable re-formulations of AR as a classification task have been proposed. They are in turn sketched by the following schemata:[8]

(6) Antecedent$_1$ Antecedent$_2$ Anaphor **1/2**

(7) Antecedent Anaphor **Y/N**

Connolly et al. ([2]) suggested instances consisting of an anaphor and two antecedent candidates, the target information left to be learnt being which of these two candidates is "better" for the anaphor in question. This information could be then utilized by a step-by-step elimination of the less plausible candidates to determine the correct antecedent.

The other formulation of the task has been proposed by McCarthy and Lehnert ([11]) and has been used by most of the state-of-the-art systems as the standard one. It postulates instances formed by an anaphor–antecedent candidate pair together with the information whether the candidate is a

[7] The mutual information scores have been computed based on dependency triples extracted from a large parsed corpus. [8] A word represents a set of features (of the entity hinted by its meaning), symbols in bold represent possible values of the target feature.

valid antecedent of the anaphor or not. This attribute determines whether the instance is understood as positive or negative.

Most ML-based AR systems use knowledge-poor features to describe the individual instances. Unsurprisingly, this poses problems similar to the ones described in the previous section. In my opinion, an important additional problem is that the features are considered out of context. The individual instances provide a very detailed description of the relationship between the anaphor and the antecedent, which is very advantageous for nominal coreference resolution, where the relation between the referred entities plays a more important role than context. However, this view of the task is very unsuitable for grasping pronominal anaphora, where different types of information, such as salience or interplay with other antecedent candidates, play an important role. Moreover, this seems to be yet a bigger issue for Czech, where, compared to English, information structure is not as tightly connected with the syntactic structure of the sentence.

One solution to this problem is introducing new features reflecting salience. In this respect, Ng and Cardie ([14]) have used the result of a syntactic search AR algorithm as a binary feature, and Preiss ([16]) has engaged the salience factors proposed in the rule-based system of Kennedy and Boguraev ([5]). The latter is a very plausible approach, for Czech with a big potential of benefiting from the rich interaction between syntax and information structure. Moreover, I would suggest re-computing the salience model iteratively during the classification phase to account for the information in already resolved links.

Another solution to this problem can be possibly obtained by a different formulation of AR as a classification task. It can be argued that the AR task has inherently the following structure:

(8) Antecedent$_n$ ... Antecedent$_1$ Anaphor **1/.../n**

Nevertheless, this concept is not very suitable for ML in this form. The main problems lie in data sparseness and the correct linearization of the antecedent candidates – these can be arbitrarily embedded into each other. On the other hand, this formulation of the AR task contains more information about the relevant context, and the information corresponding to the target feature is actually the piece of information we aim to learn – which antecedent to choose for a given anaphor from a list of candidates. The potential of this AR task reformulation needs to be investigated empirically.

4 Conclusion

In this paper, I have discussed the most notable limitation of most state-of-the-art AR systems – the fact that they disregard higher-level cues, even though these are known to play an important role. I have proposed possible ways of taking advantage of higher-level information available in the AR process, namely considering verbal valency constraints and predicate-arguments statistics. I have also suggested several ways of adapting the ML-based AR methods in order to account for the structure of the AR task more closely.

Acknowledgments

This work has been partly supported by the Ministry of Education of CR within the Center of basic research LC536.

References

1. Baldwin, B.: Cogniac: High precision coreference with limited knowledge and linguistic resources. In: Proceedings of the ACL '97/EACL '97 workshop on Operational factors in practical, robust anaphora resolution. (1997).
2. Connolly, D., Burger, J.D., Day, D.S.: A machine learning approach to anaphoric reference. In: Proceedings of the International Conference on New Methods in Language Processing (NeMLaP), ACL (1994).
3. 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) 107–115.
4. 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) 339–352.
5. Kennedy, C., Boguraev, B.: Anaphora for everyone: pronominal anaphora resoluation without a parser. In: Proceedings of the 16th conference on Computational linguistics, Morristown, NJ, USA, ACL (1996) 113–118.
6. Kilgarriff, A., Rychlý, P., Smrž, P., Tugwell, D.: The sketch engine. In: Proceedings of the Eleventh EURALEX International Congress. (2004) 105–116.
7. Lappin, S., Leass, H.J.: An algorithm for pronominal anaphora resolution. Computatinal Linguistics **20**(4) (1994) 535–561.
8. Lin, D.: Automatic retrieval and clustering of similar words. In: COLING-ACL. (1998) 768–774.
9. Linh, N.G.: Návrh souboru pravidel pro analýzu anafor v českém jazyce. Master's thesis, Charles University, Faculty of Mathematics and Physics, Prague (2006).
10. Lopatková, M., Žabokrtský, Z., Benešová, V.: Valency lexicon of czech verbs VALLEX 2.0. Technical Report 34, UFAL MFF UK (2006).
11. McCarthy, J.F., Lehnert, W.G.: Using decision trees for coreference resolution. In: Proceedings of the 14th International Conference on Artificial Intelligence IJCAI-95, Montreal, Canada (1995) 1050–1055.
12. Mitkov, R., Evans, R., Orăsan, C.: A new, fully automatic version of mitkov's knowledge-poor pronoun resolution method. In: Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2002), Mexico City, Mexico (February 17–23, 2002).
13. Němčík, V.: Anaphora resolution. Master's thesis, Masaryk University, Faculty of Informatics, Brno (2006).
14. Ng, V., Cardie, C.: Improving machine learning approaches to coreference resolution. In: Proceedings of the 40th Annual Meeting of the ACL. (2002) 104–111.
15. Pala, K., Smrž, P.: Building Czech WordNet. **2004**(7) (2004) 79–88.
16. Preiss, J.: Machine learning for anaphora resolution. Technical Report CS-01-10, University of Sheffield, Sheffield, England (Aug 2001).
17. Strube, M., Ponzetto, S.: WikiRelate! Computing semantic relatedness using Wikipedia. In: Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06), Boston, Mass. (July 2006) 1419–1424.
18. The Czech National Corpus (2006) `http://ucnk.ff.cuni.cz/english/`.

Part III

Text Processing Tools

Manatee/Bonito – A Modular Corpus Manager

Pavel Rychlý

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

Abstract. A *corpus* is a large collection of texts in electronic form. *Corpus managers* are tools or sets of tools for coping with corpora. They can encode, query, and visualise texts. This paper describes widely used corpus manager Manatee that has many unique features: modular design, dynamic attributes, multi-values, multi-language support, almost unlimited corpus size and others.
The second part of the paper presents Bonito – the graphical user interface of the Manatee system. Other extensions of the system are also mentioned.

1 Introduction

Text Corpora play a crucial role in current linguistics. They provide empirical data for studies on how a language is used. Corpora are stored in computers but there are not many applications which can handle huge corpora of size in billions of tokens (word forms) which are being available in last years.

An ideal general-purpose corpus management tool should implement the following features:

text preparation – conversion from various formats, encodings, etc.;
metadata management – integration of the information about the source of data, authors, topics, genre, ...
tokenization – language-dependent determination of the elementary unit accessed, usually a word;
corpus annotation – potentially ambiguous, manual and automatic tagging on morphological, syntactic, semantic and pragmatic levels;
efficient corpus storage – the storage structures should enable fast retrieval of all stored data
concordancing – retrieving text snippets matching the user's query;
computation of statistics – searching for typical patterns in data, frequency distribution of various features, co-occurrence statistics, etc.

2 Manatee

The presented corpus management system Manatee is able to deal with extremely large corpora and is able to provide a platform for computing a wide range of lexical statistics. It has all the above mentioned features and it is also language and tag-set/annotation independent.

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

The Manatee system is based on the text indexing library FinLib [1] that provides storage structures and retrieval procedures for corpus data based on an efficient implementation of inverted indexes, word compression, etc.

The system processes either the output of an external tokenizer or that of the simple internal one. There are two kinds of annotation that can be provided:

positional attributes adds linguistic information (like PoS tag, basic form) for each token,

structure annotation that denotes a structure in the text (e.g. a sentence, paragraph or document boundaries, noun or verb phrases).

There is also a special type of positional attributes – *dynamic attributes*. These are not store in the corpus explicitly, a function is declared in the corpus configuration. The function defines how to compute the value of the dynamic attribute for each particular token. There are three types of dynamic attribute usages:

1. transformation of tag values (e.g. displaying a full description of tag codes),
2. selection of partial information from an attribute (e.g. an attribute representing the gender is derived from a complex grammatical tag),
3. linking external information sources (e.g. linking a thesaurus database or morphological analyzer)

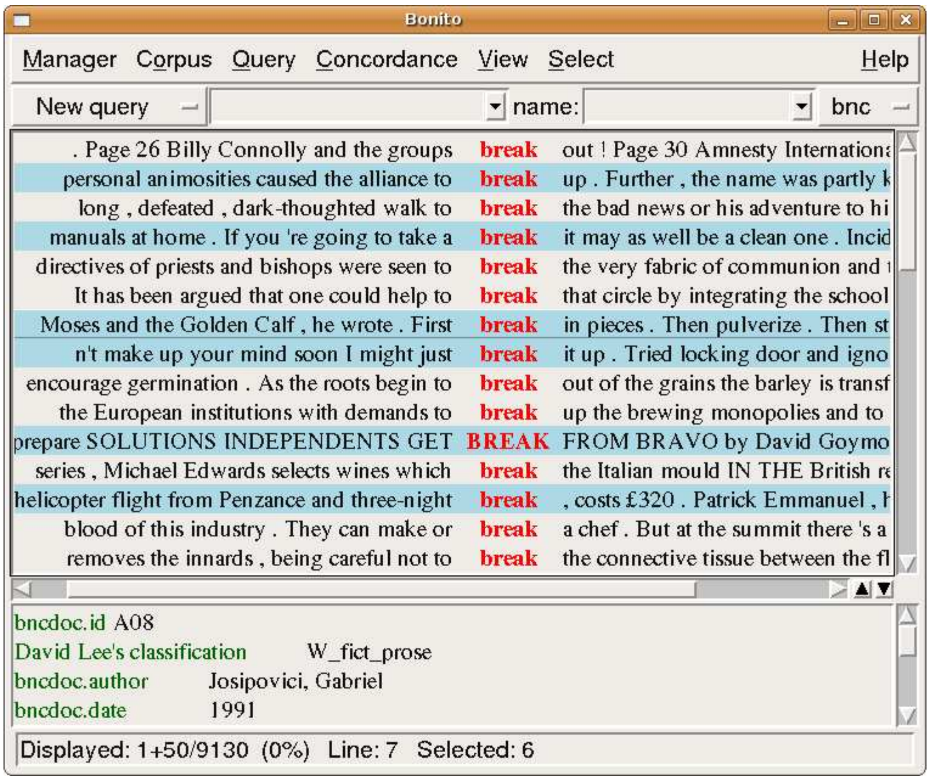
Selected attributes can be marked as ambiguous, it means the attribute can contain *multi-value* – a set of values. A user can ask for any of respective values to find such token.

The system is fully internationalized. Thextual data can be stored in different character encodings including Unicode (UTF-8 [2]) and users can define local language settings for each corpus or even for each attribute. These settings are used during query evaluation and for locale sorting conventions.

Manatee implements a powerful query language. It enables searches given by restrictions on any positional attribute, on any meta information, or on any of their combinations. A given query can be further refined by means of positive or negative filters that are applied on the current result. Meta information can be used to create sub-corpora. The query language is an extension of the popular CQP [3].

3 Bonito

Bonito is a graphical user interface (GUI) of the Manatee corpus manager. It enables queries to be formed and given to various corpora. The results are clearly displayed and can be changed in various ways. Statistics can also be computed on them.



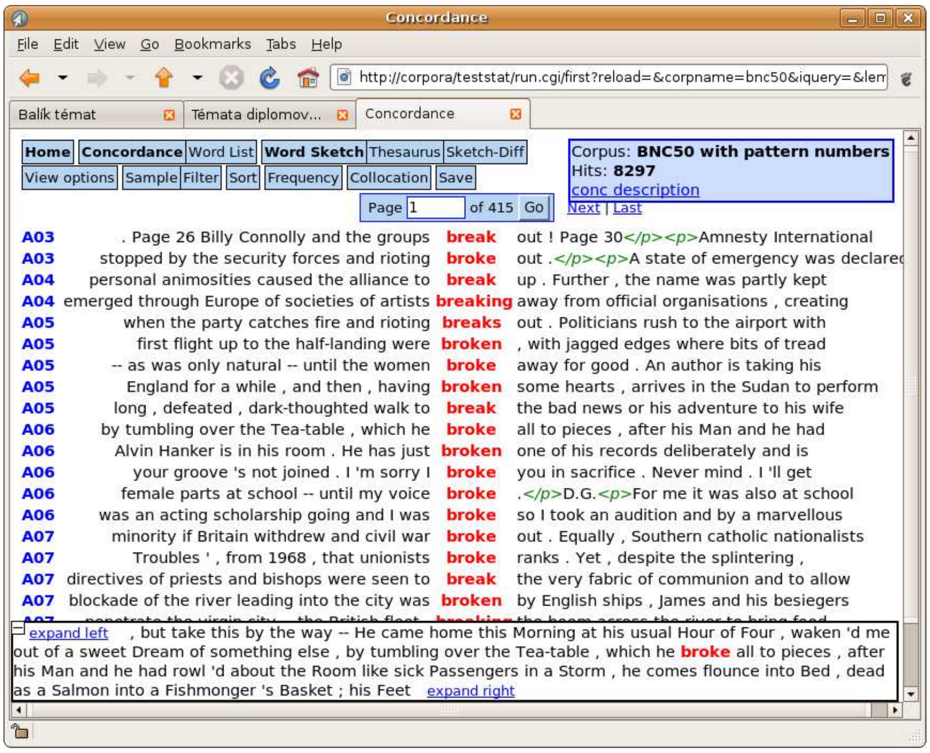
```
      . Page 26 Billy Connolly and the groups  break  out ! Page 30 Amnesty Internation
   personal animosities caused the alliance to  break  up . Further , the name was partly k
         long , defeated , dark-thoughted walk to  break  the bad news or his adventure to hi
       manuals at home . If you 're going to take a  break  it may as well be a clean one . Incid
    directives of priests and bishops were seen to  break  the very fabric of communion and t
       It has been argued that one could help to  break  that circle by integrating the school
       Moses and the Golden Calf , he wrote . First  break  in pieces . Then pulverize . Then st
         n't make up your mind soon I might just  break  it up . Tried locking door and igno
   encourage germination . As the roots begin to  break  out of the grains the barley is transf
     the European institutions with demands to  break  up the brewing monopolies and to
prepare SOLUTIONS INDEPENDENTS GET  BREAK  FROM BRAVO by David Goymo
    series , Michael Edwards selects wines which  break  the Italian mould IN THE British re
helicopter flight from Penzance and three-night  break  , costs £320 . Patrick Emmanuel , h
       blood of this industry . They can make or  break  a chef . But at the summit there 's a
     removes the innards , being careful not to  break  the connective tissue between the fl

bncdoc.id A08
David Lee's classification     W_fict_prose
bncdoc.author        Josipovici, Gabriel
bncdoc.date          1991

Displayed: 1+50/9130 (0%) Line: 7 Selected: 6
```

Fig. 1. Bonito application window

Bonito use the client/server architecture. Bonito is the client part, it is a standalone applications which runs on most systems (Unix + X Window, Windows 95+/NT/2000 ME/Vista, Macintosh and others with the Tcl/Tk suport). The server part runs on a server (Unix, Windows, ...) where corpus data are stored. The client communicates to server over the Internet (TCP/IP) connection, the communication protocol is very lightweight, even a modem connection is sufficient for proper work. It is also possible to use "local" connection without Internet, in such case, client and server runs on a PC and corpus data are stored on a local disk.

The corpus query result is the so-called concordance list that creates all corpus positions corresponding with the query given. The concordance list is then displayed in KWIC (Key Word(s) In Context) format. The searched words are displayed with their contexts one below the other. The concordance list is sometimes abbreviated as concordance.

A concordance is one of the central objects of Bonito. Most of the Bonito's window area is formed by the concordance list where query results are displayed. An example of the Bonito window is Figure 1.



Corpus: **BNC50 with pattern numbers**
Hits: **8297**
conc description

```
A03           . Page 26 Billy Connolly and the groups  break    out ! Page 30</p><p>Amnesty International
A03          stopped by the security forces and rioting  broke    out .</p><p>A state of emergency was declared
A04        personal animosities caused the alliance to  break    up . Further , the name was partly kept
A04  emerged through Europe of societies of artists breaking away from official organisations , creating
A05             when the party catches fire and rioting  breaks   out . Politicians rush to the airport with
A05             first flight up to the half-landing were  broken   , with jagged edges where bits of tread
A05              -- as was only natural -- until the women  broke    away for good . An author is taking his
A05              England for a while , and then , having  broken   some hearts , arrives in the Sudan to perform
A05             long , defeated , dark-thoughted walk to  break    the bad news or his adventure to his wife
A06            by tumbling over the Tea-table , which he  broke    all to pieces , after his Man and he had
A06             Alvin Hanker is in his room . He has just  broken   one of his records deliberately and is
A06              your groove 's not joined . I 'm sorry I  broke    you in sacrifice . Never mind . I 'll get
A06              female parts at school -- until my voice  broke    .</p>D.G.<p>For me it was also at school
A06           was an acting scholarship going and I was  broke    so I took an audition and by a marvellous
A07            minority if Britain withdrew and civil war  broke    out . Equally , Southern catholic nationalists
A07                Troubles ' , from 1968 , that unionists  broke    ranks . Yet , despite the splintering ,
A07    directives of priests and bishops were seen to  break    the very fabric of communion and to allow
A07    blockade of the river leading into the city was  broken   by English ships , James and his besiegers
```

expand left , but take this by the way -- He came home this Morning at his usual Hour of Four , waken 'd me out of a sweet Dream of something else , by tumbling over the Tea-table , which he **broke** all to pieces , after his Man and he had rowl 'd about the Room like sick Passengers in a Storm , he comes flounce into Bed , dead as a Salmon into a Fishmonger 's Basket ; his Feet expand right

Fig. 2. Bonito2 concordance page

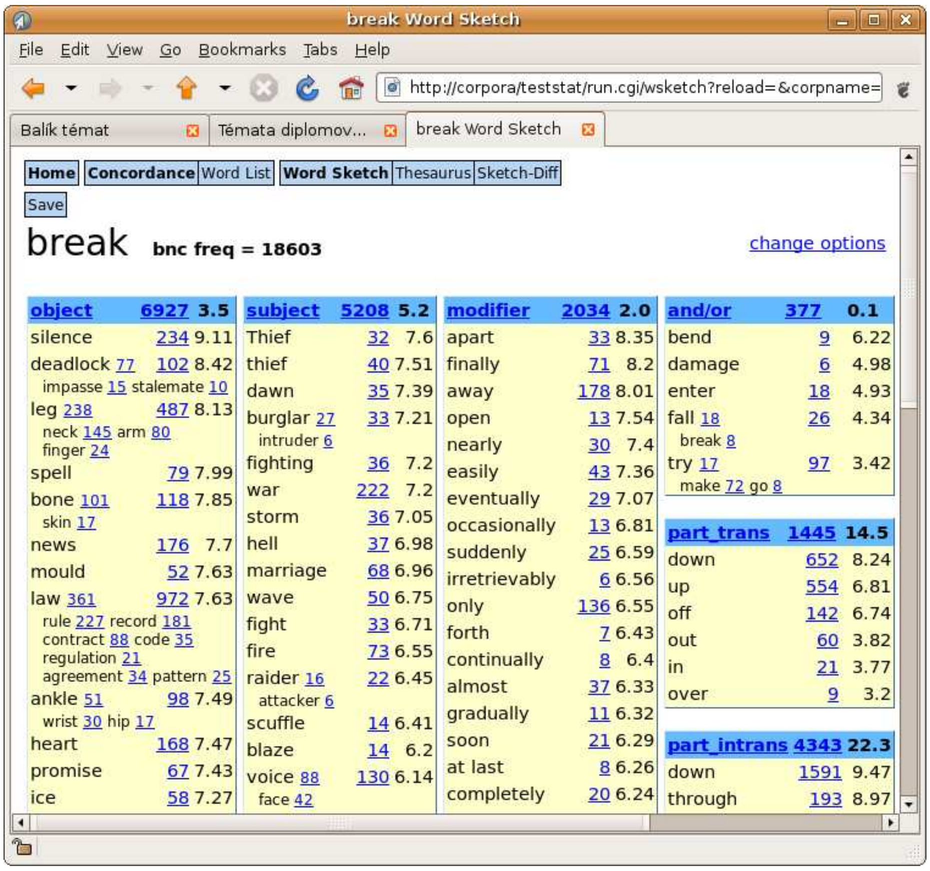
4 Bonito2

Bonito2 is a new GUI, it provedes the same (or better) functionality as the older Bonito. The crucial difference is that is a web application. Users do not need to install any client application, they use standard web browsers to access the interface. All regular web browsing techniques like cut&paste and bookmarks are available for users.

Web pages are generated by a CGI script on the web server. Standard web server authentication can be used to limit access. The known solutions of access and load control can be applied to the web server and the standard secured web protocol (https) can be used too. Finally, it is easy to connect Bonito2 to other sources and/or applications – the web pages can be read from other applications, and links to external sources (a picture, a sound sample, a video) can be presented. Concordance view is displayed on Figure 2.

5 Sketch Engine

Because of the modular design of the system, it is easy to add more functions to the system or use the system as part of another application.



break **bnc freq = 18603**

object	**6927**	**3.5**
silence	234	9.11
deadlock 77	102	8.42
impasse 15 stalemate 10		
leg 238	487	8.13
neck 145 arm 80 finger 24		
spell	79	7.99
bone 101	118	7.85
skin 17		
news	176	7.7
mould	52	7.63
law 361	972	7.63
rule 227 record 181 contract 88 code 35 regulation 21 agreement 34 pattern 25		
ankle 51	98	7.49
wrist 30 hip 17		
heart	168	7.47
promise	67	7.43
ice	58	7.27

subject	**5208**	**5.2**
Thief	32	7.6
thief	40	7.51
dawn	35	7.39
burglar 27	33	7.21
intruder 6		
fighting	36	7.2
war	222	7.2
storm	36	7.05
hell	37	6.98
marriage	68	6.96
wave	50	6.75
fight	33	6.71
fire	73	6.55
raider 16	22	6.45
attacker 6		
scuffle	14	6.41
blaze	14	6.2
voice 88	130	6.14
face 42		

modifier	**2034**	**2.0**
apart	33	8.35
finally	71	8.2
away	178	8.01
open	13	7.54
nearly	30	7.4
easily	43	7.36
eventually	29	7.07
occasionally	13	6.81
suddenly	25	6.59
irretrievably	6	6.56
only	136	6.55
forth	7	6.43
continually	8	6.4
almost	37	6.33
gradually	11	6.32
soon	21	6.29
at last	8	6.26
completely	20	6.24

and/or	**377**	**0.1**
bend	9	6.22
damage	6	4.98
enter	18	4.93
fall 18	26	4.34
break 8		
try 17	97	3.42
make 72 go 8		

part_trans	**1445**	**14.5**
down	652	8.24
up	554	6.81
off	142	6.74
out	60	3.82
in	21	3.77
over	9	3.2

part_intrans	**4343**	**22.3**
down	1591	9.47
through	193	8.97

Fig. 3. Word Sketch of verb 'break'

The Sketch Engine [4] use the whole Manatee system and Bonito2 interface and provides word sketches, grammatical relations, and a distributional thesaurus as additions. A word sketch (see Figure 3) is a one-page, automatic, corpus-derived summary of a word's grammatical and collocational behaviour. Each word sketch contains direct links to concordances illustrating the listed collocation.

6 Conclusion

This paper describes main features of the Manatee corpus management system including graphical user interfaces Bonito and Bonito2. The system is in regular use in many research groups on universities around the world. There are also commercial companies (especially publishers) which use the system or its parts or extensions in day-to-day works.

Acknowledgements

This work has been partly supported by the Academy of Sciences of Czech Republic under the projects T200610406, T100300419 and by the Ministry of Education of CR within the Centre of basic research LC536 and National Research Programme 2C06009.

References

1. Rychlý, P.: Corpus managers and their effective implementation. Ph.D. thesis, Faculty of Informatics, Masaryk University (2000).
2. Yergeau, F.: RFC2279: UTF-8, a transformation format of ISO 10646. Internet RFCs (1998).
3. Schulze, B.M., Christ, O.: The CQP User's Manual. (1996).
4. Kilgarriff, A., Rychlý, P., Smrž, P., Tugwell, D.: The Sketch Engine. Proceedings of Euralex (2004) 105–116.

Corpus Query System Bonito
Recent Development

Vojtěch Kovář

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

Abstract. This paper presents two of the new features of corpus query system Bonito: 1. Saving the outputs of the system in the XML format and 2. Localization mechanism used to enable easy translation of the system into different languages. In both cases, the developing process is described and examples of the new functionality are given. In the first sections, we also outline the general system functionality and features.

1 Introduction

At the present time, large text corpora form an important source of liguistic information. They are used for a wide variety of tasks, e.g. language learning and teaching, testing of automatic text processing tools, discovering of real words behaviour and many more lingustic research purposes. As the corpus linguistics becomes more and more popular, there is a need of good corpus query systems (CQS) that enable people to work with large text data comfortably. According to the variety of users needs, there are more and more features and functions of these CQSs needed.

At Masaryk University in Brno, a corpus manager Manatee/Bonito [1] is being developed, that is able to perform wide variety of tasks including e.g. fast searching in big corpora, computing word sketches, thesaurus and many more statistical characteristics. The system is used by researchers and lexicographers from all over the world. In order to fulfill different users needs, we continually extend the system by adding new functions.

In this paper, two of these new functions are discussed. Firstly, we briefly describe the Manatee/Bonito system in general. In the next sections the new features – saving outputs in the XML format and localization mechanism of the system – are introduced. We describe the development process of both new features and show examples of the new functionality.

2 The Manatee/Bonito System

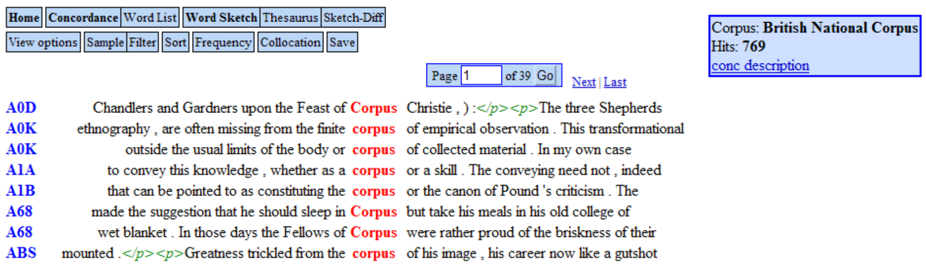
The Manatee/Bonito corpus query system consists of two parts.

Manatee provides low-level access to corpus data, it integrates fast searching in corpora and evaluation of complex queries implementing a powerful corpus query language. It also functions as a corpus management server.

Bonito serves as an interface between low-level Manatee functions and the user. Version 1 is a standard multiplatform application that connects to the Manatee server and mediates most of its functions in a user-friendly way. The newer version Bonito 2 (see Figure 1) is completely web-based. Web pages are generated on the server (using CGI), with a standard web browser serving as the corpus client.

Bonito 2 is written in Python, object-oriented and very transparent programming language. It enables the system to be well maintained and easily extensible. For generating web pages, a templating engine is used which enables easy changes in web pages appearance.

The functions desribed bellow were implemented within the scope of the Bonito 2 system.



Corpus: **British National Corpus**
Hits: **769**
conc description

A0D	Chandlers and Gardners upon the Feast of	Corpus	Christie ,) :</p><p>The three Shepherds
A0K	ethnography , are often missing from the finite	corpus	of empirical observation . This transformational
A0K	outside the usual limits of the body or	corpus	of collected material . In my own case
A1A	to convey this knowledge , whether as a	corpus	or a skill . The conveying need not , indeed
A1B	that can be pointed to as constituting the	corpus	or the canon of Pound 's criticism . The
A68	made the suggestion that he should sleep in	Corpus	but take his meals in his old college of
A68	wet blanket . In those days the Fellows of	Corpus	were rather proud of the briskness of their
ABS	mounted .</p><p>Greatness trickled from the	corpus	of his image , his career now like a gutshot

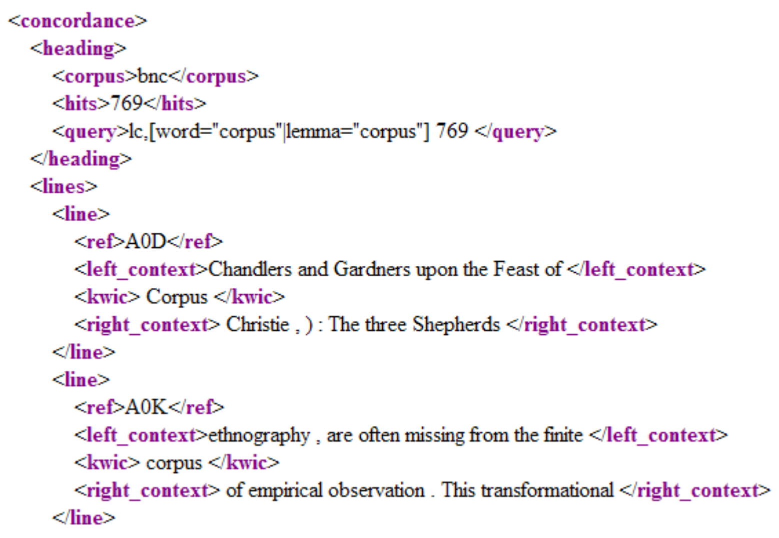
Fig. 1. Illustration of the Bonito 2 user interface

3 Saving Outputs in XML

In the Bonito 2 system, all possible outputs (concordances, word sketches, thesaurus) were in the form of HTML pages. This format is very suitable for viewing the search results but it is not very comfortable e.g. for saving and further processing of the results.

For this reason, we decided to implement two saving possibilities – plain text (in form of columns delimited by the tab character) and XML, that is currently very popular and suitable for further straightforward processing.

As mentioned in the Introduction, the system performs wide variety of tasks. Most important of them are concordance view, word lists, frequency lists, word sketches, thesaurus and collocation candidates computation. For each of these functions, there is an output in form of HTML page, realized by particular template.



```
<concordance>
  <heading>
    <corpus>bnc</corpus>
    <hits>769</hits>
    <query>lc,[word="corpus"|lemma="corpus"] 769 </query>
  </heading>
  <lines>
    <line>
      <ref>A0D</ref>
      <left_context>Chandlers and Gardners upon the Feast of </left_context>
      <kwic> Corpus </kwic>
      <right_context> Christie , ) : The three Shepherds </right_context>
    </line>
    <line>
      <ref>A0K</ref>
      <left_context>ethnography , are often missing from the finite </left_context>
      <kwic> corpus </kwic>
      <right_context> of empirical observation . This transformational </right_context>
    </line>
```

Fig. 2. Concordance – XML structure example

For saving options, we created a new set of templates that is used for text and XML output instead of HTML pages. For each output type, we designed different XML structure. The tags used in the structures are quite simple and self-explaining but they also provide a good structuralization of the data (see Figure 2). For each function, we also created a web form that enables users to modify saving options. An example of "Save Concordance" form can be seen in the Figure 3.

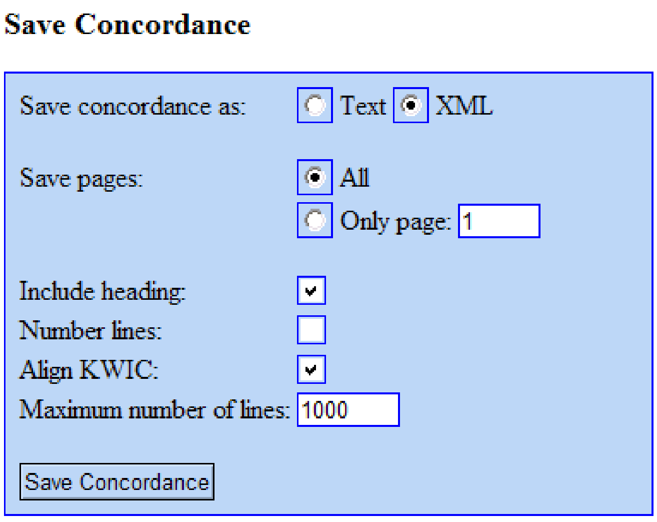
4 Localization Mechanism

The second function we have implemented is the localization mechanism. The main motivation for this step was the fact that the program is used worldwide and for many people the default English version can be non-intuitive and confusing. In the following, we describe all steps leading to well working localization mechanism.

4.1 Templating Engine

As a first step, we changed the used templating engine. The old templating engine was very tiny and simple, but it was not very fast and it also provided only small support for `gettext` utilities that we planned to use for implementing the localization mechanism (see below).

For this reason, we switched to the Cheetah Templating Engine [2], a robust templating engine based on the Python language. It has also quite intuitive


Save Concordance

Save concordance as:	○ Text [✓ ● XML]
Save pages:	[✓ ● All]
	○ Only page: 1
Include heading:	✓
Number lines:	☐
Align KWIC:	✓
Maximum number of lines:	1000

Fig. 3. Save Concordance form

syntax similar to common Python code. All templates used in the system were translated into the Cheetah language.

4.2 The Translation

For the localization itself, we used the `gettext` services integrated in the Python language[1].

In the template files, all translatable strings were replaced by `gettext` statements. By the `gettext` tools, we can now extract all translatable strings from the templates and add next localization language only by adding one file (containing traslated strings) into the system. This is very flexible, so that the system is now easily extensible.

4.3 Language Selection

Another question was how to select the correct user interface (UI) language for particuar user.We solved it by defining two possible ways of how to do that.

By default, the UI language is set according to the preferred language in the user's web browser (we got it by parsing the "HTTP_ACCEPT_LANGUAGE" parameter sent by the browser). The second possibility for the users is to associate their user name with a particular UI language. Currently, both ways are implemented.

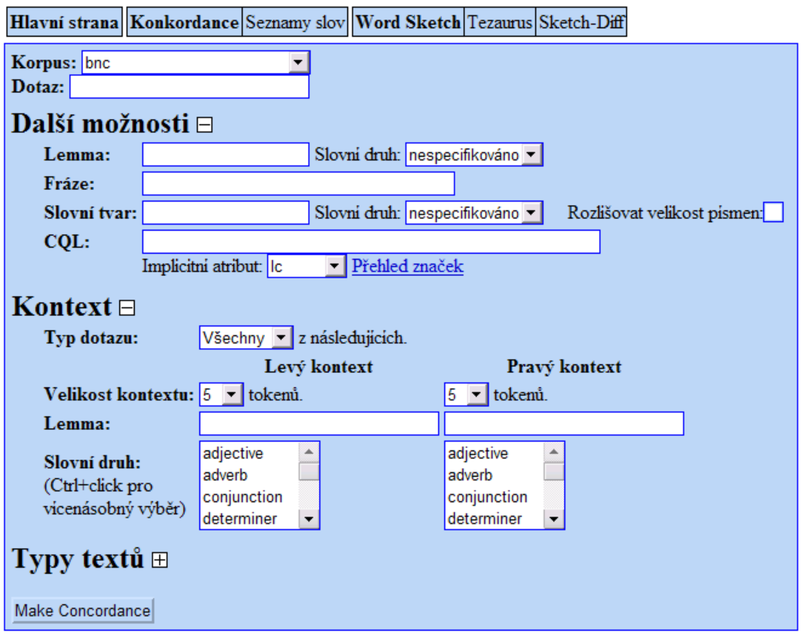
[1] `http://docs.python.org/lib/module-gettext.html`

4.4 Input and Output Encoding

When working with corpora in different languages and system with different localizations, there is a question: In what encoding should be the results presented? So far, encoding of the currently selected corpus was used in all system outputs. However, this is useless when using different localizations, e.g. Czech localization could not be used at the same time as an English corpus in ISO Latin 1 encoding.

The only possible solution seems to be using UTF 8 encoding for all outputs. This step brings particular complications, such as recoding of all outputs from selected corpus encoding into UTF 8 and all inputs from UTF 8 to the corpus encoding, but it is the only possibility to assure that the localization will work correctly.

By the input recoding, we also have to handle unknown characters (e.g. when recoding "ž" from UTF 8 into ISO Latin 1). We solved this problem by replacing unknown characters by the fullstop that matches any character in regular expressions used in the corpus query language.



Korpus: bnc
Dotaz:

Další možnosti

Lemma: Slovní druh: nespecifikováno
Fráze:
Slovní tvar: Slovní druh: nespecifikováno Rozlišovat velikost písmen:
CQL:
Implicitní atribut: lc Přehled značek

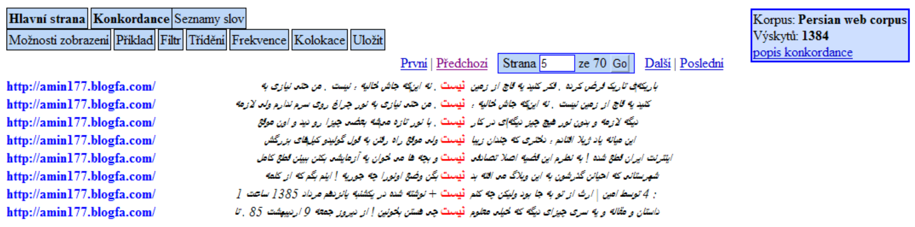
Kontext

Typ dotazu: Všechny z následujících.

	Levý kontext	Pravý kontext
Velikost kontextu:	5 tokenů.	5 tokenů.
Lemma:		
Slovní druh: (Ctrl+click pro vícenásobný výběr)	adjective, adverb, conjunction, determiner	adjective, adverb, conjunction, determiner

Typy textů

Fig. 4. The main input form in the Czech localization



http://amin177.blogfa.com/ باریکه‌ی تاریک فرض کرده . فکر کنید یه قاچ از زمین نیست . نه اینکه جاش خالیه ، نیست . من حتی نیازی به
http://amin177.blogfa.com/ کنید یه قاچ از زمین نیست . نه اینکه جاش خالیه ، نیست . من حتی نیازی به نور چراغ روی سرم ندارم ولی لازمه
http://amin177.blogfa.com/ دیگه لازمه و بدون نور هیچ چیز دیگه‌ای در کار نیست . با نور تازه همیشه بعضی چیزا رو دید و اون موقع
http://amin177.blogfa.com/ این میانه یاد ژیلا افتادم ، دختری که چندان زیبا نیست ولی موقع راه رفتن به لول گولیدو کپل‌های بزرگش
http://amin177.blogfa.com/ اینترنت ایران قطع شده ! به نظرم این قضیه اصلا تصادفی نیست و بچه ها می خوان یه آزمایشی بکنن ببینن قطع کامل
http://amin177.blogfa.com/ شهرستانی که احیانا گذرشون به این وبلاگ می افته بد نیست بگن وضع اونورا چه جوریه ! اینم بگم که از کلمه
http://amin177.blogfa.com/ : 4 توسط امین | ارث از تو به جا بود ولیکن چه کنم نیست + نوشته شده در یکشنبه پانزدهم مرداد 1385 ساعت 1
http://amin177.blogfa.com/ داستان و مقاله و یه سری چیزای دیگه که خیلی معلوم نیست چی هستن بخونین ! از دیروز جمعه 9 اردیبهشت 85 . تا

Fig. 5. Persian corpus and Czech localization

4.5 The Czech Localization

A a sample, we created a Czech localization of the system. The user interface in Czech is illustrated in the Figure 4. In the Figure 5, the corpus of Persian is shown within the system with Czech localization.

5 Conclusions and Future Directions

In the paper, we have presented two of recently added features in the Manatee/Bonito corpus query system. We described motivations, development process and some problems related to the implementation as well as their solutions.

In the future development, we want to add more features to enable more comfortable work with the system. Corpora are very valuable source of linguistic information and we want users from all over the world to be able to exploit their benefits.

Acknowledgements

This work has been partly supported by the Academy of Sciences of Czech Republic under the project T100300414.

References

1. Rychlý, P., Smrž, P.: Manatee, Bonito and Word Sketches for Czech. In: Proceedings of the Second International Conference on Corpus Linguisitcs, Saint-Petersburg, Saint-Petersburg State University Press (2004) 124–132.
2. Rudd, T., Orr, M., Bicking, I.: Cheetah: The python-powered template engine (2004) `http://www.cheetahtemplate.org/Py10.html`.

Morphemic Analysis: A Dictionary Lookup Instead of Real Analysis

Pavel Šmerk

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

Abstract. This paper presents an approach for developing morphological and morphemic analysis systems for inflective languages based on a simple and fast dictionary lookup instead of any kind of analysis of the input word form. This approach allows the information about the word forms (lemma, tag, morpheme structure, derived words, derivational relations) to be described according to the traditional grammars' models and to have such a description completely independent of any requirement of the analysis process.

1 Introduction

According to Gelbukh and Sidorov [5], the designer of a morphological analyzer for an inflective language has the following choice:

- either generate all word forms and build a system with a large dictionary and a very simple "analysis" (just searching) algorithm,
- or build a system with a much smaller dictionary of stems with information about possible endings, but with some more sophisticated algorithm (analysis through generation, in particular).

For the inflective languages they strongly suggest the second option, because it allows to use a grammar model almost directly taken over from traditional grammars, which are oriented mainly toward generation. These traditional models are rather simple, but foremost intuitive for a system developer or morphological database editors.

The aim of this paper is to show that it is possible to preserve all advantages of the use of traditional grammars' models retaining quite simple "analysis" by word forms dictionary searching algorithm as well. And, moreover, this applies not only to the morphological analysis, but even to the morphemic analysis, i.e. formal description of the derivational morphology. The suggested approach is based on the use of Jan Daciuk's [2] algorithms and tools for construction of minimal deterministic acyclic finite state automata (DAFSA) and on his (but not only his, cf. further) idea of how to use such automata for morphological analysis.

Actually, the basic ideas of all what follows were published by Jan Daciuk and other authors almost ten years ago. It is hard to believe that there does not seem to be any real world implementation in the time being, at least for Slavic languages, for which it could be very interesting due to their high inflectionality. There are some experimental implementations, of course, but not any really used Slavic inflectional or even derivative morphology based on Daciuk's DAFSAs or on any similar approach.

The following section illustrates the basic idea, why it is advantageous to represent a morphological dictionary by the minimal DAFSA. It does not get stuck into technical details concerning the actual construction of the minimal DAFSA. The algorithms are published, and even ready-to-use tools are available.[1] Instead of it, in Section 3, shows the possible arrangements of the data for morphological and morphemic analysis and guessing. Section 4 discusses some further advantages of the proposed approach.

2 Basic Idea

The basic idea is very simple, but very powerful. Any finite list of unique strings can be considered to be a finite (formal) language and as such can be represented by some DAFSA. If we choose the minimal one, then a partial path corresponding to any left or right substring shared by some subset of the modelled strings will occur exactly once in such minimal automaton.[2]

For example, let us consider the following fraction of some dictionary of the word form and morphological tag pairs[3]:

```
Kanaďánek:c1nSgMk1
Kanaďánka:c2nSgMk1
Kanaďánkovi:c3nSgMk1
Holanďánek:c1nSgMk1
Holanďánka:c2nSgMk1
Holanďánkovi:c3nSgMk1
```

One can see that all pairs with the word forms of the same lemma (lexeme) *Kanaďánek* or *Holanďánek* share some same left substring, in particular *Kanaďán* and *Holanďán* respectively. Similarly, all pairs with the nominative, genitive and dative singular of the masculine animate nouns share at least the same right substring *:c1nSgMk1*, *:c2nSgMk1* and *:c3nSgMk1* respectively — but in the case of the same declension or derivational type they can share a few characters

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

[2] To be precise, this is not true in some of the cases, when in the same string some left substring shared by more strings overlap some right substring, which is also shared by more strings. But due to the regularity of an absolute majority of words in a natural language it does not affect the following claims dramatically (cf. some results for Nonslavic languages at the beginning of the following section).

[3] All examples in this paper are in Czech: *Kanaďánek* is a little Canadian boy (diminutive), *Holanďánek* is a little Dutch boy. The text after the colon is a morphological tag: c1, c2 and c3 is the first, the second and the third case (i.e. nominative, genitive and dative), nS is number – singular, gM is gender – masculine animate, k1 is part of speech – noun.

more: in the data above they share even *ďánek:c1nSgMk1*, *ďánka:c2nSgMk1* and *ďánkovi:c3nSgMk1*. Moreover, all pairs of this example share e.g. the *nSgMk1* right substring. Since the inflective languages tend to express all morphological categories at the end of the word form (and, moreover, often in one single ending, but it is not important for now), we can take for granted that similar relations will hold through the whole list of all word form and tag pairs.

The analysis based on an automaton which encodes a list of the word form and tag pairs is quite straighforward. If an analysed string concatenated with the separator (the colon in the case above — it serves as the sign of the end of the string to distinguish for example *Kanaďánka* and *Kanaďánkama*[4]) is found in the automaton then each possible remaining path to the final state of the automaton encodes one of the possible morphological tags. The “searching" of the string is as simple (and therefore also quick, of course) as possible, because the automaton is deterministic, so that for each possible string there is at most one straightforward path without need of any backtracking.

Apparently, the creation of such a minimal DAFSA can be a simple way to dramatically reduce the size of the list of word form and tag pairs (or any other similar list) and thus to get rid of the disadvantage of the list's big size. Moreover, the creator of the list has a full freedom of choice of the manner in which he or she creates the list. The method used will have no influence on the effectiveness or complexity of the proces of analysis, because the management of the data is completely separated from their usage.

There is only one condition which has to be met to get really compact representation of the dictionary: strings representing the data must not have any unique parts. The next section shows how to arrange the data for various parts of the morphological and morphemic analysis to meet this condition.

3 Data Representation

As was shown above, the list of word form and tag pairs is suitable for the construction of the DAFSA. But a serious problem arises when we want to include the information about the lemma:

```
Kanaďánek:Kanaďánek:c1nSgMk1
Kanaďánka:Kanaďánek:c2nSgMk1
Holanďánek:Holanďánek:c1nSgMk1
Holanďánka:Holanďánek:c2nSgMk1
```

Obviously, all combinations word+lemma and lemma+tag are unique among all strings. DAFSA can be constructed, of course, but it would be too large. The solution used by e.g. Daciuk [2], Kowaltowski [8] or in a slightly different manner by the INTEX project [10] is simple: store only the right substring of the lemma, in which it differs from the particular word form:

[4] colloquial form of the instrumental plural

```
Kanaďánek::c1nSgMk1
Kanaďánka:Bek:c2nSgMk1
Holanďánek::c1nSgMk1
Holanďánka:Bek:c2nSgMk1
```

where the *B* as the second letter of the aplhabet means "to get the lemma delete two last characters from the word form and then attach the *ek*". It is clear now, that the list in this form has the same properties as the list of word form and tag pairs in previous section and as such can be "compressed" to a minimal DAFSA. The lookup is similar as was above: after finding the word form, the remaining paths to the final state describe all proper lemma+tag combinations. It should be noted that the compression rate could be very high. Kowaltowski [8] reports 0.25 byte per one word-tag-lemma entry for Brazilian Portuguese and Daciuk [2] reports less then 0.15 byte per one word-lemma-tag entry for German and ca. 30 times better compression rate compared to gzip on that data. Quite preliminary results for Czech data show similar compression potential.

The following subsections are only variations on this solution.

3.1 Generating All Word Forms from the Lemma

This is quite simple. All we need is to swap word form and lemma in order to create list of lemma:word form:tag triples. Instead of the full word form there is, of course, only an ending part in which the word form and lemma differ. The generation is similar to the above: the lemma is looked up and the remaining paths describe all possible word forms with proper tags. To generate all word forms from a word form other then lemma it is the best option to analyse the input word form to get the lemma and then look up the rest.

3.2 Morphemic Analysis

Let us suppose we have a mechanism, which can handle productive derivational suffixes and derive words according to some rules. Then we derive all word forms we can (maybe except for some recursive rules as for great-great-grandmother and so on) and track the history of the derivation. Of course we track the internal, "deep" forms of morphemes before any phonological rules or morphophonological or stem internal alternations apply. Then a dictionary may look as follows:

```
Kanaďánek:Ed-an-0k-~
Holanďánek:Ed-an-0k-~
Kanaďánka:Ed-an-0k-a
Holanďánka:Ed-an-0k-a
```

where the *E* as the fifth letter in the alphabet means "strip last five characters", as above. The first segment, *d*, serves for recovering the root morpheme (whose last phoneme has been palatalised, so the base form is *Kanad-* as English *Canad-*), the following segments are derivational suffixes and the last one is the inflectional ending.[5]

Obviously, such a morphemic "analysis" is as simple as the morphological analysis above. It constrasts with Zeldes's [12] very recent attempt to morphological/morphemic[6] analysis of Polish resulting in the analysis algorithm which is "computationally more complex, but lexicographically more compact alternative to text-based morphological analysis techniques currently in use for Polish"

3.3 Generating All Derived Words

To generate derived words we need a list of stems with possible suffixes[7] encoded in the usual way:

```
Kanad:Aďan
Kanaďan:Bánek
Holand:Aďan
Holanďan:Bánek
```

The interpretation (generation) has to be (or can be) recursive, e.g. *Kanad-* $\longrightarrow$ *Kanaďan* $\longrightarrow$ *Kanaďánek*.

Another list is needed for determining the word which the analysed word was derived from:

```
Kanaďan:Cd
Kanaďánek:Dan
Holanďan:Cd
Holanďánek:Dan
```

It allows recursive interpretation as well. It is sufficient to have both these lists created for the lemmata only, not for all word forms.

3.4 Morphological and Morphemic Guessing

The dictionaries for guessing morphological or morphemic characteristics of unknown words are a bit different. There are no full word forms in them, but only the "surface" forms of endings or endings with sequences of suffixes, possibly followed by the alternated root final consonant(s). This whole potential right substring of an unknown word is reverted and the analysed words are matched from their ends. It is of a high importance to use only really productive endings and suffixes to avoid the overgeneration. The first list is for the lemma and tag guessing:

[5] *0* stands for vowel *e* alternating regularly with zero and ˜ stands for zero ending — of course, all these signs are chosen completely arbitrarily. [6] Zeldes wants to get a linguistically adequate split of an analysed word to the stem and ending, he does not perform a full morphemic analysis

[7] Yes, there are also prefixes in the language, let us put them aside, as handling them would be technically more difficult, but it would not bring any new idea.

```
kenáď::c1nSgMk1
aknáď:Bek:c2nSgMk1
```

the second for the morphemic guessing:

```
kenáď:Ed-an-0k-~
aknáď:Ed-an-0k-a
```

The structures of both are the same as in the previous subsections.

This paper has started with a polemic against Gelbukh and Sidorov [5]. The same two argued a year before [4] that (only) the algorithmic, non-dictionary approaches to the morphological analysis allow the guessing of the unknown word forms. It should be clear now, that even "dictionary" approach allows the same.

4 Advantages of the Proposed Approach

There are certainly no doubts that natural language processing needs to have a usable and reliable derivational (and *really* derivational, e.g. not only static description of derivational relations between lexicon entries) morphology.

Then the main advantage of the proposed approach arises from the fact, that one has to describe and implement all morphophonological alternations or at least large part of them to accomplish really good derivational morphology (and morphemic analysis as well). The "every exception is a new paradigm"-like approaches (e.g. for Czech language analyser `ajka` [9]) or Jan Hajič's analyser [6]) hardly can be successful, because they are too redundant from the derivation's point of view, and therefore too complex regarding the maintainance of the data. Consider, for example, that a need of addition of some colloquial noun endings appears: having several hundreds noun paradigms, it will lead either in a creation of some ad hoc and thus possibly erroneous scripts, or in an manual update of the most frequent paradigms only. Both cases can cause inconsistencies in the data[8].

On the other hand, a system with a small number of paradigms which computes[9] these morphophonological (or even stem or root internal, if someone wants to distinguish it) alternations in the realtime (i.e. during the analysis phase) has and has to have too complex code to maintain as well. Or, such a system is too complex at least compared to the approach proposed in this paper, which is: however complex description of the data you have or want to have, for the analysis generate all of them in advance and then use only simple static dictionaries.

Even the well known two-level morphology [7] leads to superfluous complexity and unevincible linguistic inadequateness (comparing to described approach) when used for Slavic languages. Either we have to have a list of stem alternations (which are rather frequent in these languages). But such a solution

[8] This is one of the real present-day deficiencies of the `ajka` analyser. [9] This "computes" means some kind of algorithm more complex than a simple walk through some FSA or FST.

resigns to any adequate description of the regularity of these alternations. Or we can encode these alternations into the phonological rules which are likely to be linguistically inadequate. And if someone wants to use some prepared tools like Xerox's `xfst` [1][10], either has to develop some translation from his or her data description to the formal language that the particular tool uses, or is bound to that language.[11]

Thus, the main advantage is the complete separation of the "analysis" process and the description of the data. This separation allows not only the free choice of the model for morphological and morphemic description of the data (namely of the possible morpheme combinations and the alternations caused by these combinations), but it also to a great extent simplyfies changes of this description when needed.

Very simple example: let us suppose some set of words which belong to a particular paradigm, and another set of words which all have some same additional form (e.g. due to some diachronnic reasons), so that the analyzer distinguishes two separate paradigms differing only by this one extra form. Now let us imagine, that we would like to lower the redundancy of our paradigm system by employing some inheritance principles. We would want to have one base paradigm and one derived from this base one by adding the extra form. To achieve this in traditional models of analysis[12], we would have to not only modify the tools managing the database of lexemes and paradigms, but also the analyzer. And the second could be not so simple as the analyzers are in general optimized for high speed of processing and low size of the data. Using the approach proposed in this paper, the proper modification of the database managing tools suffices in such case and the analyzer itself may remain intact.

4.1 Description of Productivity

The described approach is very handy especially for the description of the regularities in the language, namely the process of the derivation of new or infrequent words. For instance, most of newly created or taken over Czech verbs adopt the suffix *-ov* so that the lemma ends with *-ovat*, e.g. *programovat*[13]. But there is a whole bunch of suffixes completely regularly connected with this type of verbs: *-ován, -ovaný, -ování, -ovaně, -ovanost, -ovatelný, -ovatelně, -ovatelnost*[14]

Adding a verb from this class to a dictionary of word form, lemma and tag triples adds in the DAFSA only the stem or some its part and one arc to the common beginning of all these suffixes. But the language of the DAFSA is by such an addition enriched with all word forms generated from lemmas, which means about 150 word forms (many of them are homonymous, however). Of

[10] But unfortunately `xfst` is not freely available. [11] Of course, it is possible to process some part of derivational morphology in this way, as [11] did for Czech. In her paper only three rules are shown — and all of them are awkward if one imagine that whole derivational morphology should be done in this "write-only" manner.

[12] at least in all really used ones we are aware of [13] to programme [14] Taking the verb to programme, the glosses can be something like: programmed (pass part), programmed (adj), programming, ?, ?, programmability, programmable (adj), programmable(?) (adv).

course, it would be useful to have some mechanism of (some tool for) an simple adding of a new word to such a class also on the generating side of the analyzer system, i.e. the suffixes should be really interconnected somehow. However, in this is the power of this approach: one arc in the automaton is able to enrich the accepted language with even thousands of words. On the other hand, many of these c. 150 words will occur in real texts rarely or even never. And many of them or may be even all of them could be correctly analysed as unknown words. But why try to guess if we are able to know, and it worths almost nothing? On the other hand, even if we prefer guessing in some cases, it can be very simplified using the described approach.

4.2 Some Minor Advantages

– The description of morphophonological alternations needs not to be efficient, because it does not affect the process of analysis at all. It is very important, as it allows to use the description, which is really adequate for the data. Moreover, it allows the free choice of the programming language for tools managing the data, e.g. some high level and more comfortable scripting language. It also allows the scripts (or programs) to be optimized for maintainability, which is important for long-life projects.
– It allows to have either several smaller cooperating tools or several quite independent parts of a analyzer, thus the whole project would be less complex, which may prevent some programmers' mistakes from arising.
– This approach also allows a gradual move from some previous morphological analysis system. It is possible to have some parts of the derivational (or, of course, also inflective) morphology described in the new system with proper capture of important productive relations, and to take over the rest (perhaps non-productive, and therefore more tricky to describe) of the morfology from the previous system in a form of a plain list of word form, lemma and tag triples etc.

5 Conclusion

As was said at the beginning, the basic ideas of the proposed approach are not new at all, but their potential seem to be rather underestimated and maybe they worth to be recalled a bit, what was the aim of this paper.

A new morphological and morphemic analyser for Czech is developed using this approach, but still in the phase of data preparation (unification of related paradigms etc.), therefore there are no results available yet.

Acknowledgments

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 basic research LC536 and by the Czech Science Foundation under the project 201/05/2781.

References

1. Kenneth R. Beesley and Lauri Karttunen. 2003. Finite State Morphology. CSLI Publications, Stanford University. 509 p.
2. Jan Daciuk. 1998. *Incremental Construction of Finite-State Automata and Transducers, and their Use in the Natural Language Processing.* Ph.D. dissertation, Technical University of Gdańsk, Poland.
3. Jan Daciuk. 2001. *Experiments with Automata Compression.* Proceedings of Conference on Implementation and Application of Automata CIAA'2000. LNCS 2088, Springer-Verlag, pp. 105–112.
4. Alexander Gelbukh and Grigori Sidorov. 2002. *Morphological Analysis of Inflective Languages through Generation.* In: J. Procesamiento de Lenguaje Natural, No 29. Sociedad Española para el Procesamiento de Lenguaje Natural, ISSN 1135-5948, pp. 105–112.
5. Alexander Gelbukh and Grigori Sidorov. 2003. *Approach to Construction of Automatic Morphological Analysis Systems for Inflective Languages with Little Effort.* In: Computational Linguistics and Intelligent Text Processing. Proc. CICLing-2003. LNCS 2588, Springer-Verlag, pp. 215–220.
6. Jan Hajič. 2004. Disambiguation of Rich Inflection: Computational Morphology of Czech. Charles University, The Karolinum Press. 328 p.
7. Kimmo Koskenniemi. 1984. *A General Computational Model for Word-Form Recognition and Production* Proceedings of COLING-84, Stanford University, pp. 178–181.
8. Tomasz Kowaltowski, Cláudio L. Lucchesi, and Jorge Stolfi. 1998. *Finite Automata and Efficient Lexicon Implementation.* Technical Report IC-98-02, University of Campinas, São Paulo.
9. 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.
10. Max Silberztein. 1998. *INTEX 4.1 for Windows: A Walkthrough.* Proceedings of The Third International Workshop on Implementing Automata, WIA'98. LNCS 1660, Springer-Verlag, pp. 230–243.
11. Hana Skoumalová 1997 *A Czech Morphological Lexicon* Proceedings of the Third Meeting of the ACL Special Interest Group in Computational Phonology, pp. 41–47.
12. Amir Zeldes. 2006. *Abstracting Suffixes: A Morphophonemic Approach to Polish* Proceedings of KONVENS 2006 (Konferenz zur Verarbeitung natürlicher Sprache), Universität Konstanz.

Part IV

Lexical Semantics

Classification of Multilingual Mathematical Papers in DML-CZ

Preliminary Excursion

Petr Sojka, Radim Řehůřek

Masaryk University, Faculty of Informatics, Brno, Czech Republic
`sojka@fi.muni.cz`, `xrehurek@fi.muni.cz`

Abstract. The growth of digital repositories of scientific documents is speed-ed up by various digitisation activities. Almost all papers of mathematical journals are reviewed by either Mathematical Reviews or ZentralBlatt Math, summing up to more than 2.000.000 entries.
In the paper we discuss possibilities and experiments we did on the data of Czech Digital Mathematics Library, DML-CZ with the goal of developing novel scalable methods of document classification and retrieval of multilingual mathematical papers.

1 Motivation – Project of Digital Mathematics Library

You always admire what you really don't understand. (Blaise Pascal)

Mathematicians from all over the world dream of World Digital Mathematics Library [1], where (almost) all of reviewed mathematical papers in all languages will be stored, indexed and searchable with the today's leading edge information retrieval machinery. A good resources towards this goals–in addition to the publisher's digital libraries–are twofold:

1. 'local' repositories of digitised papers as NUMDAM [2][1], DML-CZ [3][2] or born-digital archives CEDRAM [4][3]), arXiv.org>math[4]
2. two review services for the mathematical community: both ZentrallBlatt Math[5] and Mathematical Reviews[6] have more than 2.000.000 entries (paper metadata and reviews) from more than 2300 mathematical serials and journals.

Google Scholar[7] is becoming useful in the meantime, but lacks specialised math search and metadata guessed from parsing crawled papers are of low quality (compared to the controlled repositories).

Both review services agreed on the supported Mathematics Subject Classification (MSC) scheme[8], and currently used MSC 2000 is being revised for use in

[1] `http://www.numdam.org` [2] `http://www.dml.cz` [3] `http://www.cedram.org`
[4] `http://arxiv.org/archive/math` [5] `http://www.zblmath.fiz-karlsruhe.de/MATH/`
[6] `http://www.ams.org/mr-database` [7] `http://scholar.google.com` [8] `http://www.ams.org/msc/`

2010 (MSC2010). Most journals request classification being used already by authors when submitting journals for publication; however, most of retrodigitised papers published before MSC 1990 are not classified by MSC in the databases.

Within the DML-CZ project we have investigated possibilities to classify (retrodigitised) mathematical papers by machine learning techniques, to enrich math searching capabilities and to allow semantically related search. As text of scanned pages is usually optically recognised, machine learning algorithms may use not only metadata (and reviews, if any), but also full text. Interesting question to pose is to find to which extent mathematical formulae are important for classification, document similarity measures, and search.

2 Data Preprocessing

We run carelessly to the precipice, after we have put something before us to prevent us seeing it. (Blaise Pascal)

There are many modelling techniques for given classification task in the area of pattern recognition. To design a classifier, we have to choose measurable features. These features should be as discriminative as possible with regard to the pattern of interest. Most of the methods use bag of words representation of a document. There are methods such as Latent Semantic Analysis (LSA), that try to find main document topics based on word co-occurences in documents.

2.1 Primary data

The data available for experiments are metadata and full texts of mathematical journals covered by DML-CZ project. During the first three years of the project, we have digitized and collected data in digital library, accessible via web tool called Metadata editor[9]. To date (November 2007), in the digitised part there are 351 volumes of 9 journals: 1449 issues, 11725 articles on 173779 pages. We are promised to get another 15000+ full texts of articles of other digitisation project. In addition, digital born data of currently processed articles by various journals are being imported into the library, as workflow of paper publishing process was modified a bit so that all fine-grained metadata including the full text are exported for the digital library for long-term storage (CEDRAM).

By 2009, we target for a digital library with about 50000 mathematical articles with full texts, and much more with basic article metadata (abstracts, reviews).

For first experiments, we have used two types of data:

1. texts from scanned pages of digitized journals (usually before 1990, where no electronic data are available);
2. texts from 'digital-born' papers, written in TEX.

[9] `editor.dml.cz`

We started our experiments with retrodigitised articles from the *Czech Mathematical Journal* (CMJ)[10] from years 1951 to 1991 (starting 1992 there exist born-digital data). We took only those papers where both primary MSC classification in Zentrallblatt and Mathematical Reviews agree. This was done to ensure a clean training and evaluation set. In addition, we have used only part of the text corpus of the journal: only MSC categories with more than 60 papers were trained in the experiments. We got 925 papers in eight MSC categories:

class 05-xx (Combinatorics): 129 articles
class 06-xx (Order, lattices, ordered algebraic structures): 178 articles
class 08-xx (General algebraic systems): 64 articles
class 20-xx (Group theory and generalizations): 147 articles
class 34-xx (Ordinary differential equations): 146 articles
class 46-xx (Functional analysis): 70 articles
class 53-xx (Differential geometry): 87 articles
class 54-xx (General topology): 104 articles

Second text corpora we used in our experiments was created from papers of Journal *Archivum Mathematicum*[11] from years 1992–2007, where we had TEX source files available. For machine learning we use MSC categories for which we had at least 40 papers – they were categories 34, 53 and 58 (Global analysis, analysis on manifolds).

2.2 Preprocessing and methods used

It is widely known that design of the learning architecture is very important, as is preprocessing, learning methods and their parameters [5].

First part of the preprocessing is tokenizing the input documents. We used alphabetic tokenizer, with lowercase or Krovetz stemmer [6]. No stoplists were used, no word bi-grams, no lemmatization yet.

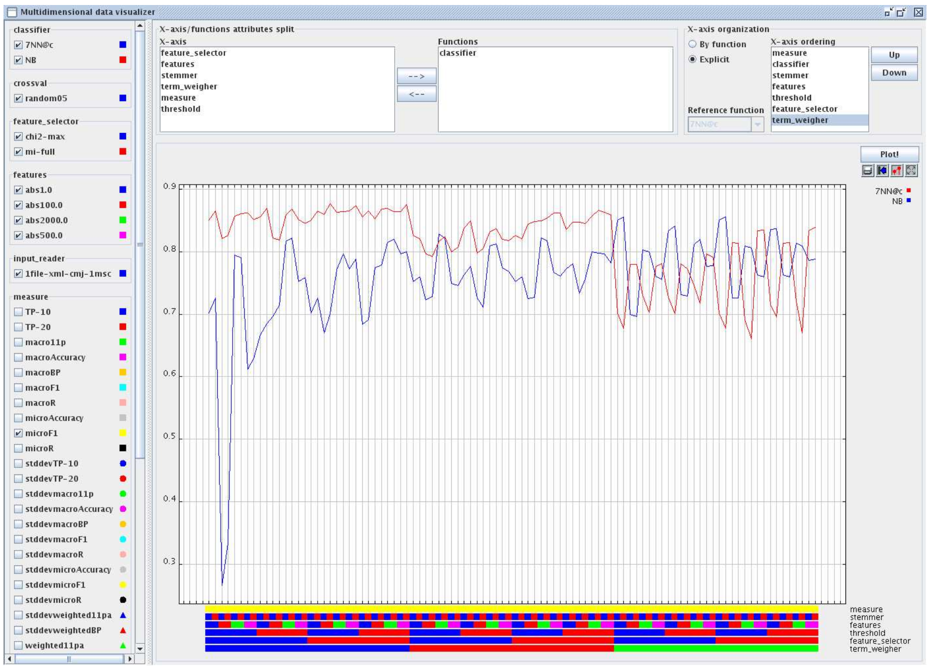
The setup of the experiments is such that we run whole bunch of training attempts in multidimensional learning space of learning methods, features, term weighting types and classifiers:

feature selectors: χ^2, mutual information
feature amount: 100, 500, 2000, all features
term weighting: *bnn*, *nnn*, *atc* [7] (corresponding to binary, term frequency and augmentented TF*IDF weighting schemes in SMART notation)
threshold estimators: fixed, s-cut
classifiers: Naive Bayes, Artificial Neural Network (six hidden units, threshold function tanh), *k*-Nearest Neighbours and Support Vector Machines

For evaluation purposes, we take note on micro/macro F1, TP10, TP20, 11-point-average, accuracy, correlation coefficient, break-even point and their standard deviations for 10-fold crossvalidation.

[10] `http://cmj.math.cas.cz/` [11] `http://www.emis.de/journals/AM/`

All these results are then compared to see which 'points' in the parameter space perform best. This framework allows easy comparison of the evaluated parameters with visualization of the whole result space – see for example multidimensional data visualization on Figure 1. For details see [5].



Multidimensional data visualizer

classifier
- [✓] 7NN@c 🟦
- [✓] NB 🟥

crossval
- [✓] random05 🟦

feature_selector
- [✓] chi2-max 🟦
- [✓] mi-full 🟥

features
- [✓] abs1.0 🟦
- [✓] abs100.0 🟥
- [✓] abs2000.0 🟩
- [✓] abs500.0 🟪

input_reader
- [✓] 1file-xml-cmj-1msc 🟦

measure
- [ ] TP-10 🟦
- [ ] TP-20 🟥
- [ ] macro11p 🟩
- [ ] macroAccuracy 🟪
- [ ] macroBP 🟧
- [ ] macroF1 🟦
- [ ] macroR 🟥
- [ ] microAccuracy ⬜
- [✓] microF1 🟨
- [ ] microR ⬛
- [ ] stddevTP-10 ●
- [ ] stddevTP-20 ●
- [ ] stddevmacro11p ●
- [ ] stddevmacroAccuracy ●
- [ ] stddevmacroBP ●
- [ ] stddevmacroF1 ●
- [ ] stddevmacroR ●
- [ ] stddevmicroAccuracy ●
- [ ] stddevmicroF1 ●
- [ ] stddevmicroR ●
- [ ] stddevweighted11pa ▲
- [ ] stddevweightedBP ▲
- [ ] weighted11pa ▲

X-axis/functions attributes split

X-axis	Functions
feature_selector	classifier
features	
stemmer	
term_weigher	
measure	
threshold	

X-axis organization
- By function
- [✓ Explicit]

Reference function: 7NN@c

X-axis ordering: measure, classifier, stemmer, features, threshold, feature_selector, term_weigher

Fig. 1. Framework for comparing learning methods. This figure shows comparison of the *k*NN and Naive Bayes classifiers. On the horizontal axis there are particular combinations of the learning space parameters and on the vertical axis the microaveraged F1 measure.

3 Preliminary Results

We know the truth, not only by the reason, but also by the heart. (Blaise Pascal)

Apart from classification, we also tried Latent Semantic Analysis (LSA) [8] to see which concepts are the most relevant.

3.1 Language is relevant

There were papers in several different languages in the CMJ data. After listing the top concepts in LSA of CMJ it is clear that the most significant concepts correspond to language:

1. 0.3*"the" +0.19*"and" +0.19*"is" +0.18*"that" +0.15*"of" +0.14*"we" +0.14*"for" +0.11*"ε" +0.11*"let" +0.11*"then"
2. −0.41*"ist" −0.40*"die" −0.28*"und" −0.26*"der" −0.23*"wir" −0.21*"für" −0.17*"eine" −0.17*"von" −0.14*"mit" −0.13*"dann"
3. −0.31*"de" −0.30*"est" −0.29*"que" −0.27*"la" −0.26*"les" −0.2*"une" −0.2*"pour" −0.20*"et" −0.18*"dans" −0.18*"nous"
4. −0.36*"цхто" −0.29*"для" −0.23*"пусть" −0.19*"из" −0.19*"если" −0.16*"так" −0.16*"то" −0.14*"на" −0.14*"тогда" −0.131169*"мы"
5. −0.33*"semigroup" −0.25*"ideal" −0.19*"group" −0.18*"lattice" +0.18*"solution" +0.16*"equation" −0.16*"ordered" −0.15*"ideals" −0.15*"semigroups" −0.13*"prime"
6. 0.46*"graph" +0.40*"vertices" +0.36*"vertex" +0.23*"graphs" +0.2*"edge" +0.19*"edges" −0.18*"ε" −0.15*"semigroup" −0.13*"ideal" +0.13*"connected"
7. 0.81*"ε" −0.25*"semigroup" −0.16*"ideal" +0.12*"lattice" −0.11*"semigroups" +0.10*"i" −0.1*"ideals" +0.09*"ordered" +0.09*"ř" −0.08*"idempotent"
8. 0.29*"semigroup" −0.22*"space" +0.2*"ε" +0.19*"solution" +0.19*"ideal" +0.18*"equation" +0.16*"oscillatory" −0.15*"spaces" −0.16*"compact" +0.14*"ds"
9. 0.28*"lattice" −0.27*"ε" +0.27*"ordered" +0.23*"group" −0.21*"semigroup" +0.2*"subgroup" −0.19*"ideal" −0.18*"space" +0.16*"groups" +0.16*"torsion"
10. −0.57*"tolerance" −0.22*"compatible" −0.21*"congruence" −0.20*"tolerances" +0.19*"ideal" +0.16*"group" +0.14*"subgroup" +0.13*"prime" −0.13*"algebras" −0.13*"algebra"

First concepts clearly capture the language of the paper (EN, DE, FR, RU), and only then topical itemsets start to be grabbed. It is not surprising – the classifiers then have to be trained either for every language (there is sparsity problem for languages as Czech, Italian or German even French presented in the digital library), or the document features have to be chosen in a language independent manner by mapping words to some common topic ontology. To the best of our knowledge, nothing like EuroWordNet for mathematical subject classification terms or mathematics exists.

3.2 Math notation may be relevant

We also ran LSA on the monolingual corpora of Archivum Mathematicum, where mathematics formulae were not thrown away (recall that this is a subcorpora created from TEX files). Again, taking note of the topmost concepts and their most significant components, we may observe that there appear a few terms containing mathematical formulae (here r and m^n):

1. −0.32*"t" −0.24*"ds" −0.17*"u" −0.17*"_" −0.17*"x" −0.15*"solution" −0.12*"equation" −0.11*"q" −0.11*"x_" −0.11*"oscillatory"

2. 0.28*"ds" +0.28*"t" −0.22*"bundle" −0.16*"natural" +0.15*"oscillatory" −0.15*"vector" +0.13*"solution" −0.13*"connection" −0.13*"manifold" +0.11*"t_0"
3. −0.22*"bundle" +0.19*"ring" −0.17*"natural" −0.16*"oscillatory" +0.15*"fuzzy" −0.15*"ds" +0.12*"ideal" −0.11*"t" −0.11*"r_0" −0.11*"nonoscillatory"
4. 0.29*"ring" −0.23*"x_" −0.21*"_" +0.21*"oscillatory" +0.18*"ideal" +0.17*"r" +0.16*"prime" +0.15*"rings" +0.13*"nonoscillatory" −0.12*"x_n"
5. −0.30*"_" −0.29*"a_" −0.17*"q_" −0.15*"ij" +0.14*"ds" −0.14*"x_" +0.14*"x_n" −0.14*"u_" +0.14*"fuzzy" +0.13*"measurable"
6. 0.87*"fuzzy" +0.19*"x_" +0.10*"oscillatory" +0.10*"ordered" −0.09*"x_n" +0.07*"nonoscillatory" +0.07*"objects" +0.07*"oscillation" −0.06*"ring" −0.06*"periodic"
7. −0.31*"ring" −0.21*"ds" −0.2*"r" −0.17*"rings" −0.17*"ideal" −0.15*"u" +0.13*"oscillatory" −0.12*"prime" +0.12*"curvature" −0.17*"x_"
8. −0.35*"ds" +0.26*"r_0" +0.2*"dx" −0.19*"t_" −0.16*"x_" +0.15*"x_n" −0.15*"holonomic" +0.14*"z" −0.13*"_" +0.12*"natural"
9. −0.24*"r_0" +0.23*"curvature" +0.16*"fuzzy" −0.15*"x_" +0.14*"symmetric" +0.13*"riemannian" +0.13*"m^n" +0.13*"connection" −0.124373*"ordered" −0.124305*"lattice"
10. 0.28*"x_" −0.25*"r_0" −0.24*"ds" −0.18*"fuzzy" +0.15*"oscillatory" +0.14*"holonomic" −0.13*"curvature" −0.12*"u" −0.11*"m^n" +0.1*"oscillation"

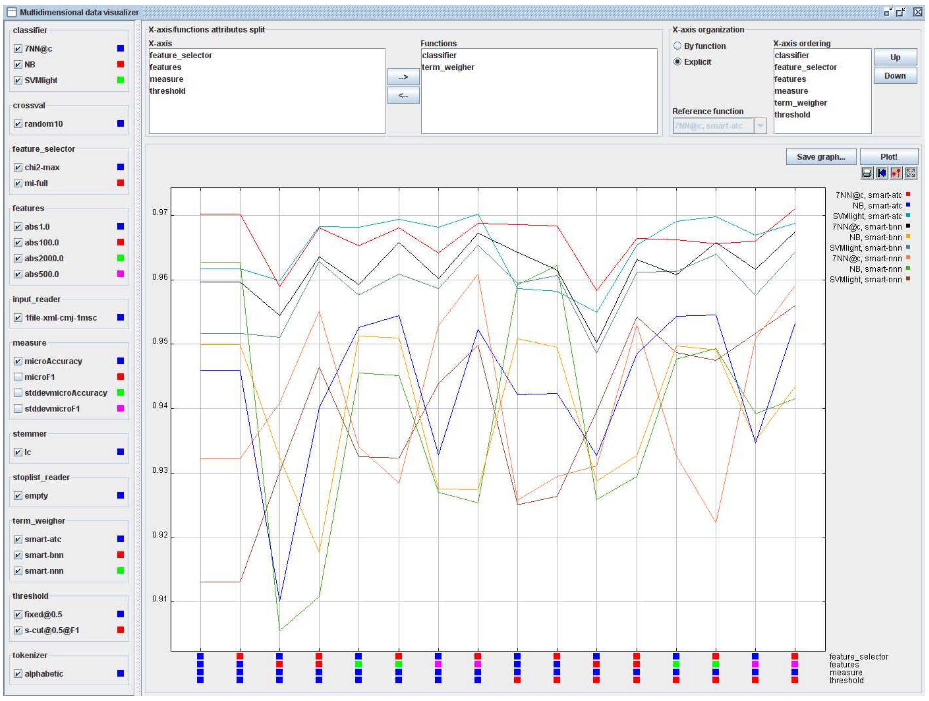
3.3 MSC classification can be learned

Detailed evaluation of classification accuracy shows that with almost all methods we easily reach about 90 % classification accuracy to classify the first two letters of primary MSC. With fine-tuning the best method (Support Vector Machine with Mutual Information feature selection, *atc* term weighting and 500–2000 features) we can increase the accuracy to 95 % or more.

4 Conclusions and Future Work

Words differently arranged have a different meaning,
and meanings differently arranged have different effects.
(Blaise Pascal)

The results presented show feasibility of machine learning approach to the classification of mathematical papers. Given enough data, when we extrapolate the results of preliminary experiments with linear machine methods (creating separable convex spaces in multidimensional feature space) we could approach very high accuracy 98 % or even more. With ambitions for even higher accuracy, higher order models (deep networks) should be used. Mainstream machine learning research was concentrated on using "convex", shallow methods (SVM, neural networks with backpropagation training) so far. State-of-the-art fine



Multidimensional data visualizer

classifier
- ☑ 7NN@c
- ☑ NB
- ☑ SVMlight

crossval
- ☑ random10

feature_selector
- ☑ chi2-max
- ☑ mi-full

features
- ☑ abs1.0
- ☑ abs100.0
- ☑ abs2000.0
- ☑ abs500.0

input_reader
- ☑ 1file-xml-cmj-1msc

measure
- ☑ microAccuracy
- ☐ microF1
- ☐ stddevmicroAccuracy
- ☐ stddevmicroF1

stemmer
- ☑ lc

stoplist_reader
- ☑ empty

term_weigher
- ☑ smart-atc
- ☑ smart-bnn
- ☑ smart-nnn

threshold
- ☑ fixed@0.5
- ☑ s-cut@0.5@F1

tokenizer
- ☑ alphabetic

X-axis/functions attributes split

X-axis: feature_selector, features, measure, threshold

Functions: classifier, term_weigher

X-axis organization: ○ By function ◉ Explicit

Reference function: 7NN@c, smart-atc

X-axis ordering: classifier, feature_selector, features, measure, term_weigher, threshold

Fig. 2. Side by side comparison of classifier and term weighting performance. Each coloured line represents performance of a combination of classifier (SVM, *k*NN or Naive Bayes) together with a term weighter (*atc*, *bnn* or *nnn*). The evaluation measure here is microaveraged accuracy, that is, the portion of correctly classified test examples. We may see that *k*NN and SVM outperform Naive Bayes and both work consistently best with the *atc* term weighting.

tuned methods allow very high accuracy even on large scale classification problems. However, training of these methods is exceptionally high and the models are big. Using the ensambles of classifiers make the situation even worse (size even bigger), and the final models need to be regularized.

Training large models with non-convex optimization [10] may give classifications that does not exhibit overfitting.

Further studies will encompass fine-grained classification trained on bigger collections, scaling issues, and fine-tuning the best performance by choosing the best set of preprocessing parameters and machine learning methods.

Acknowledgements This study has been partially supported by the grants 1ET208050513 and 1ET208050401 of Academy of Sciences Czech Republic.

References

1. Jackson, A.: The Digital Mathematics Library. Notices of the AMS **50** (2003) 918–923.
2. Bouche, T.: Towards a Digital Mathematics Library? (2006) accepted for publication as a book chapter in Communicating mathematics in the digital era (CMDE) 2006 by A.K. Peters.
3. Sojka, P.: From Scanned Image to Knowledge Sharing. In Tochtermann, K., Maurer, H., eds.: Proceedings of I-KNOW '05: Fifth International Conference on Knowledge Management, Graz, Austria, Know-Center in coop. with Graz Uni, Joanneum Research and Springer Pub. Co. (2005) 664–672.
4. Bouche, T.: A pdfLaTeX-based automated journal production system. TUGboat **27** (2006) 45–50.
5. Pomikálek, J., Řehůřek, R.: The Influence of Preprocessing Parameters on Text Categorization. International Journal of Applied Science, Engineering and Technology **1** (2007) 430–434.
6. Krovetz, R.: Viewing morphology as an inference process. In: Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Linguistic Analysis (1993) 191–202.
7. Lee, J.H.: Analyses of multiple evidence combination. In: Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Combination Techniques (1997) 267–276.
8. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by Latent Semantic Analysis. Journal of the American Society for Information Science **41** (1990) 391–407.
9. Ježek, K., Toman, M.: Documents categorization in multilingual environment. In: Proceedings of ElPub 2005, Leuven, Belgium, Peeters Publishing (2005) 97–104.
10. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H.: Greedy layer-wise training of deep networks. In Schölkopf, B., Platt, J., Hoffman, T., eds.: Advances in Neural Information Processing Systems 19. MIT Press, Cambridge, MA (2007) 153–160.

The Relations between Semantic Roles and Semantic Classes in VerbaLex

Dana Hlaváčková

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

Abstract. In this paper we present the database of verb valency frames for Czech language named VerbaLex being created presently in NLP Laboratory at Faculty of Informatics Masaryk University. This work involves building the valency database of Czech verbs with their surface and deep valency frames. Moreover we adopt the list of verb semantic classes from English to Czech. We want to show the way of more precisely subclassification of semantic classes for Czech verbs.

1 Introduction

VerbaLex – a large lexical database of Czech verb valency frames has been developed since 2005 at the Natural Language Processing Laboratory at the Faculty of Informatics Masaryk University (FI MU). VerbaLex is based on three existing independent resources:

1. BRIEF – dictionary of 50 000 valency frames for 15 000 Czech verbs is source of lexical data for VerbaLex. BRIEF was created at FI MU in 1997 [1]. The different verb senses are not distinguished here and valency frames are surface only, without any semantic information.
2. VALLEX – valency lexicon of Czech verbs is based on the formalism of the Functional Generative Description (FGD) and has been developed during the Prague Dependency Treebank (PDT) project [2]. Vallex and VerbaLex are similar projects with some important distinctions. The way of transformation of plain text format (for dictionary editing) to another formats (xml, pdf, html) used in Vallex has been used and changed for VerbaLex.
3. Czech WordNet valency frames dictionary, was created during the Balkanet project [3] and contains 1 359 valency frames (incl. semantic roles) associated with 824 sets of synonyms (synsets).

The organization of lexical data in VerbaLex comes out from the WordNet structure [4]. The lexical units in WordNet are organized into synsets arranged in the hierarchy of word meanings (hyper-hyponymic relations). For that reason, the headwords in VerbaLex are formed with lemmata in a synonymic relation (synset subsets) followed by their sense numbers (standard Princeton-WordNet notation). The *basic valency frames* (BVF) display two types of information – the constituent elements of valency frames cover both syntactic level

and lexical semantic level. The default verb position 'VERB' as the centre of the sentence is marked on the syntactic level. The pattern of sentence constituents are situated in left and right positions in accordance with the complementarity needed by the verb. The constituent elements of frame entries are entered as pure pronominal terms, e.g. kdo (who), co (what), or prepositional phrase pattern (with the lemma of the preposition) followed by the number of the required grammatical case of the phrase. This way of notation allows to differentiate an animate or inanimate subject or object position. The types of verbal complementation (nouns, adjectives, adverbs, infinitive constructions or subordinate clauses) are precisely distinguished in the verb frame notation. There is marked up the type of valency relation for each constituent element – obligatory 'obl' (must be present) or optional 'opt'. BVF is followed by simple example of usage verb in sentence. For example:
Synset: bavit:1, rozptýlit:2, rozptylovat:2
(PrincetonWordNet: amuse:2 /make (somebody) laugh)

frame: AG <person:1>$^{\text{obl}}_{\text{whoNom}}$ VERB PAT <person:1>$^{\text{obl}}_{\text{whatAccus}}$
ACT <act:2>$^{\text{opt}}_{\text{by doing whatInstr}}$

– *example: impf: bavil děti hrou (he amused the children by playing the game)*

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 (*perfective – pf., imperfective – impf. or both aspects – biasp.*);
– types of verb use (*primary – prim., figurative – fig., idiomatic – idiom.*);
– types of reflexivity for reflexive verbs.

For example:
SYNSET: BAVIT:1, ROZPTÝLIT:2, ROZPTYLOVAT:2
DEFINITION: poskytovat někomu zábavu/make (somebody) laugh

– *passive: yes*
– *meaning: I*
– *class: amuse-31.1-1*
– *impf: bavit:1 pf: rozptýlit:2 impf: rozptylovat:2*

frame: AG <person:1>$^{\text{obl}}_{\text{whoNom}}$ VERB PAT <person:1>$^{\text{obl}}_{\text{whatAccus}}$
ACT <act:2>$^{\text{opt}}_{\text{by doing whatInstr}}$

– *example: impf: bavil děti hrou (he amused the children by playing the game)*

– *attr: use: prim, reflexivity: obj_ak*

Current version of VerbaLex 2.0 contains 7 063 synsets, 23 461 verb senses, 10 596 verb lemmata and 21 100 valency frames. Valency database is available in txt, xml and html formats [5].

2 Semantic Roles

Semantic information of verb complementation is represented by two-level semantic roles in BVF. The first level contains the main semantic roles proposed on the 1stOrder-Entity and 2ndOrderEntity basis from EuroWordNet Top Ontology [6]. The 1^{st} level semantic roles represent close list of 29 semantic tags (e.g. *AG – agent, OBJ – object, INS – instrument, ACT – activity, INFO – information, SUBS – substance* etc.). On the second level, we use specific literals (lexical units) from the set of PrincetonWordNet Base Concepts with relevant sense numbers. We can thus specify groups of words (hyponyms of these literals) replenishable to valency frames. This concept allows us to specify valency frames notation with large degree of sense differentiability (e.g. *SUBS(beverage:1), OBJ(furniture:1), INS(edge tool:1)* etc.). The list of 2^{nd} level semantic roles is open, current version contains about 1 000 wordnet lexical unites.

3 Semantic Classes

We work with verb semantic classes that were originally adopted from the Levin's list of English verb classes [7] (48 classes). We also use the list of Martha Palmer's VerbNet project with more fine-grained sets of verbs [8] (82 classes, total of 395 subclasses). These verb classes have been translated and adopted for Czech language. Czech classes were enriched with Czech synonyms, aspect counterparts and Czech prefixed verbs. Presently, we work with 82 semantic verb classes, 258 subclasses and 6 393 Czech verb lemmata in the current version of our list. In building the semantic classes we prefer semantic criteria against the syntactic alternations used by Levin. As a result we get verb classes that are semantically more consistent than Levin's.

4 Relations

The process of adopting and enriching Czech semantic classes initially started with Levin/Palmer's classes but within VerbaLex we try to modify them with regard to the semantic features of predicate-argument structures of Czech verbs. Our aim is to create classes based also on the inventory of the semantic roles denoting verb arguments. This approach allows us to build semantic classes and subclasses more precisely in many cases.

Our point of view is based on assumption that verbs complemented by the identical 2^{nd} level semantic roles belong to one semantic class. For example, the verbs linked to the semantic role beverage:1 (it occurs in 42 valency frames in VerbaLex) can create following semantic groups:

beverage consumption – *pít/drink, upíjet/sip, bumbat/guggle, ochutnávat/taste...*
oversized beverage consumption – *chlastat/booze, opíjet se/soak, přihnout si/swig...*
beverage serving – *čepovat/tap, točit/draw, nalévat/pour, napojit/water...*
beverage preparation – *zkvasit/ferment, vařit/brew, ledovat/frost, protřepat/shake...*
physical result after oversized beverage consumption – *zvracet/vomit, dávit/throw up, blinkat/be sick...*

In Levin/Palmer's list of semantic classes this type of verbs belongs mostly to class 39. Verbs of Ingesting and to wide and more closely undefined class 45. Verbs of Change of State.

Verbs complemented by 2nd level semantic role *furniture:1* (it occurs in 60 valency frames in VerbaLex) can create following semantic groups:

furniture usage – *posadit se/sit down, ležet/lie, uložit se/lie down...*
furniture handling – *sklopit/recline, uklidit/tidy away, srovnat/order, umístit/place, stěhovat/move, otevřít/open...*
furniture making and maintenance – *čalounit/upholster, mořit/ebonize, leštit/polish, sklížit/glue...*

In Levin/Palmer's list of semantic classes this type of verbs belongs mostly to wide classes 9. Verbs of Putting, 45. Verbs of Change of State and 47. Verbs of Existence.

Verbs complemented by 2nd level semantic role *vehicle:1* (it occurs in 153 valency frames in VerbaLex) can create following semantic groups:

modes of movement – *zrychlit/accelerate, zpomalit/slow down, brzdit/brake, couvat/back a car, zatočit/turn, předjet/overtake...*
meet with an accident – *nabourat/smash car, narazit/crash...*
transport of people – *jet/go, nastoupit/get in, vystoupit/get out, cestovat/travel, dojíždět/commute...*
transport of load – *vézt/carry, naložit/load, vyložit/unload, přepravit/transport...*
visual and acoustic signals – *houkat/hoot, troubit/toot, blikat/blink, burácet/roar...*

In Levin/Palmer's list of semantic classes this type of verbs belongs mostly to classes 11. Verbs of Sending and Carrying, 18. Verbs of Contact by Impact, 43. Verbs of Emission and 51. Verbs of Motion.

5 Conclusion

The described type of classification can be used for 2nd level semantic role with reasonable frequency in VerbaLex (from 30 to 1 000 occurrences). The roles with general meaning and frequency higher than 1 000 occurrences are not suitable for this purpose (e.g. *object:1* – 2 500 occurrences). In spite of this, 2nd level semantic roles in VerbaLex present significant support for subclassification of verb semantic classes.

Acknowledgements

This work has been partly supported by the Academy of Sciences of Czech Republic under the projects 1ET100300414 and 1ET200610406 and by the Czech Science Foundation under the project 201/05/2781.

References

1. Pala, K., Ševeček, P.: Valence českých sloves. In: Proceedings of Works of Philosophical Faculty at the University of Brno, Brno, MU (1997), 41–54.
2. Žabokrtský, K.: Valency Lexicon of Czech Verbs. Ph.D. thesis, Prague (2005).
3. BalkaNet: Balkanet project website (2001–2004).
4. Fellbaum, C.e.: WordNet. An Electronic Lexical Database. MIT Press, Cambridge (1998).
5. Hlaváčková, D., Horák, A.: Verbalex – New Comprehensive Lexicon of Verb Valencies for Czech. In: Computer Treatment of Slavic and East European Languages, Third International Seminar, Bratislava, VEDA (2005) 107–115.
6. Vossen, P., Bloksma, L.e.a.: The EuroWordNet Base Concepts and Top Ontology. In: Technical Report Deliverable D017, D034, D036, WP5 EuroWordNet, LE2-4003, Amsterdam, University of Amsterdam (1998).
7. Levin, B.: English Verb Classes and Alternations: A Preliminary Investigation, Chicago, The University of Chicago Press (1993).
8. Palmer, M., Rosenzweig, J., Dang, H.T.e.a.: Investigating regular sense extensions based on intersective Levin classes. In: Coling/ACL-98, 36th Association of Computational Linguistics Conference, Montreal (1998) 293–300.

Keyness in Shakespeare's Plays

Jiří Materna

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

Abstract. This paper describes a novel method of identifying keyness in texts using relational learning. The relational learning is based on Inductive Logic Programing (ILP) in first-order logic. Using this method we can extract first-order logic formulas satisfied in a set of positive examples and not satisfied in a set of negative examples.
We tested this method on a collection of Shakespeare's plays to identify keyness (or aboutness) of particular plays. The research was especially related to Hamlet, Prince of Denmark which was already investigated by Mike Scott [1]. He used his own tool WordSmith, based on wordlists generating. Aim of this paper is to describe another way of automatic identifying keyness and to show that this method can find more comprehensive keyness representation.

Key words: keyword, keyness, Shakespeare, ILP

1 Introduction

In our life we often use the term 'key' to identify something important. The term is so widely used that keyness seems to be generally intuitively obvious. Here, though, we must think about the term more carefully. We should distinguish between language, mind, culture and text keyness [2]. For example, the term 'thee' can be a keyword in recent texts because of its archaic nature but it can hardly be considering a keyword in Shakespeare's language. In this paper, we are interested only in the text keyness, that is, the other aspects must be excluded. In order to eliminate language, mind or culture influence, we use a referential corpus composed of texts of the same type as an investigated text. In our case we use as a referential corpus set of all Shakespeare's plays.

First, we will describe classical methods of identifying aboutness in texts (Kintsch's and van Dijk's propositional analysis, Hoey's method) and then we will aim our effort much more precisely to clarify Scott's approach and a method of keyness extraction in first-order logic.

1.1 Keyness and aboutness

In the past, text linguists worked on related issues without using term keyness. One of the most famous approaches is Kintsch's and van Dijk's propositional analysis [3]. The method starts by splitting a text into its propositional components. For example, the sentence

```
Three big green elephants were crossing the Main Street
yesterday.
```

may have following propositions:

```
The elephants are three
The elephants are big
The elephants are green
The elephants were crossing the street
The street is called Main Street
It happened yesterday
```

It does not matter how each proposition is expressed. We are not dealing with words or clauses, we are handling concepts. It would be possible to replace each proposition with an abstract symbol or paraphrase in another language. Kintsch and van Dijk then proceed to study which of the propositions get referred to most in the entire set. That means, the method identifies the propositions which are most important in the sense that they get linked to most of all in the text [3]. This approach is called *macropropositions*. These macroproposition seem to be, more than the others, what the text is really about.

Another author who has tackled issue of aboutness in texts was Hoey [4]. His method is similar to the Kintsch's and Dijk's one. The difference is that it does not take propositions but whole sentences. Like Kintsh and van Dijk, Hoey seeks out the elements which are most linked. A link for Hoey is based on the plain text before him. What he counts as a link is a repetition of some kind. It need not be only a verbatim repetition but for example grammatical variants like the same lemma, synonym, hyponym or meronym.

2 Keywords and Scott's Keyness Analysis

The method of identifying keyness used by Scott is based on keywords. Keyword is defined like a word form that is frequent in an investigated text. Repetition here is a simple verbatim repetition, so we don't consider terms 'car' and 'cars' to be the same token.

Simple verbatim repetition alone is not, however, a good indicator of what is important and what a text is about. It is obvious that the most frequent terms will be determiners like 'the' or 'of', verbal auxiliaries and words usually occurring in general texts. These terms can hardly be good indicators of aboutness [5]. What we are looking for are terms like 'Romeo', 'Juliet', 'love', 'death', 'poison' etc. in example of Rome and Juliet.

To eliminate unwanted frequent terms, we often use a referential corpus. The referential corpus should be a set of general texts in the same language and style as an investigated text. We simply compute frequent terms for both investigated and referential corpus and exclude terms frequented in both ones.

To do this, Scott uses his own text processing tool called WordSmith [6]. This tool is based on wordlist computing. Wordlist is a list of text tokens paired with their frequency in corpus. The most useful way of arranging this list is to sort it by frequency, so you can easily filter the infrequent items. The threshold frequency which determines what terms will be considered to be a keyword is usually established experimentally.

In his work, Scott defines keyness as a set of related keywords. He noticed that keywords can be globally spread or locally concentrated in the text, so he was interested in collocational neighbors of each keyword in the text. If there are other keywords nearby, in terms of keyness, they are qualified to be a key together. The important issue is, of course, a span. In his experiments, Scott uses narrow span (1 to 5 tokens) and wide span (11 to 25 tokens). It was demonstrated that wide span rather tends to identify genre keyness, whereas the narrow span is more suitable for text keyness investigation.

3 Keyness in Terms of Relational Learning

In contrast to previous method, relational learning express keyness not only by a set of keywords but it can represent the aboutness or concept by relations between words, their attributes or positions, and even between document segments like sentences, paragraphs or phrases. For us, the key is then a relational pattern which is frequent throughout the document.

Formally, following [7], keyness K is a set of logic formulas in first-order predicate logic with modal operators that are frequent for the document. We call such formulas frequent patterns [8]. A frequent pattern is a conjunction of predicates from a given set of predicates called background knowledge. This set must contain a keyword predicate `keyword/2`. `keyword(D, KW)` predicate holds if `KW` is a keyword for the document `D`.

Background knowledge then consists of relations that are domain independent (e.g. describing relative position of words like `before/3`, `after/3`, `follows/3`, `precedes/3`, their modal variants (e.g. `always_after/3`, `always_before/3`) or that, describing morphological and syntactical categories. E.g. `hasVerb(Sentence, Subject, Verb, Object)` returns for a given sentence a triple (subject, verb, object). Background knowledge can also contain domain dependant predicates that express semantics of a word. An example is information about synsets or hypo/hyperonymic relations obtained from domain dependent ontology.

Each frequent pattern is characterized by a level of significance. A level of significance is given by a number of instances, typically a set of words and their attributes that are covered by this formula. This level of significance is called support.

An example of a frequent pattern is below.

```
word(S, B), after(S, B, C), begCap(S, C),
hasTag(S, C, 'NNP'), after(S, C, D), hasTag(S, D, 'CC')
```

It says that ″in the sentence A, there is a word B, somewhere on the right there is a word C which starts with a capital letter and has tag ′NNP′ and somewhere right from the word C there is a word D with tag ′CC′″.

4 An Experiment

We tested this method on a collection of Shakespeare's plays. In our interest was especially Hamlet, Prince of Denmark. Our main goal was try to find some additional information about keywords found by Scott.

He defines two kinds of keywords – positive and negative. Positive keywords are the ones that are outstandingly frequent in the text and negative keywords are outstandingly infrequent comparing to referential corpus.

In Hamlet, he found following positive keywords:

```
SENSE, VERY, DEATH, LORD, DO, IT, MOST, LET, WOO'T,
PHRASE, THE, T, COULD, E'EN, WHY, OF, A, OR, THIS,
WHERETO, HOW
```

Negative keywords for all Shakespeare's plays are listed below:

```
A, AND, DOTH, FATHER, FOR, FROM,GOOD, HE, HER, HIM,
HIS, I, I'LL, IN, IT, KING, LORD, LOVE, MASTER, ME,
MOST, MY, OF, OUR, SHE, SIR, THE, THEE, THEIR, THERE,
THY, THOU, TIS, WE, WHAT, WHY, YOU, YOUR, DO
```

You can see that some keywords are unsurprising. For example ′death′ is supposed to be a keyword in play about death. But what is surprising, among positive keywords there are words like ′do′ or ′it′. These tokens are even negative keywords in the other plays. There is no idea why these words should be a positive key in Hamlet and relatively insignificant in the other Shakespeare's plays. Scott's tool does not provide any feature to solve this issue, so we tried to answer it using our method.

In order to ease our task we only selected four words to be analyzed:

```
DO, IT, LORD, MOST
```

From both Hamlet corpus (HC) and all Shakespeare's plays corpus (AC), we only selected the sentences containing one of the Scott's keyword (for each keyword separately). Each created document was then splitted into sentences. It implies that we did not take into account any relations that concern two or more sentences. These sentences were then tagged by Memory-Based Shallow Parser [9] on morphological and syntactical level. Information about characters and position in the play for each sentence was added too.

When the data were prepared, we employed relational learning system RAP [10]. RAP is an ILP system for mining maximal frequent patterns in first-order logic written in Prolog. It uses generate-test paradigm for finding first-order clauses which cover at least *N* examples. The value *N* is so called minimal support and supposes to be given by user.

The most important issue in relational learning is background knowledge definition, that is, issue of what predicates can a frequent pattern consist of. Firstly, we tried to use following predicates:

`key(S)` – in the document, there is a sentence S
`hasWord(S, W)` – somewhere in the sentence S, there is a word W
`before(S, A, B)` – in the sentence S, there is a word A before B
`isWord(W, T)` – the word W has a token T
`pers(S, P)` – the sentence S is pronounced by P

Even though we set maximal pattern length to 7 there were no longer patterns than 6 found. It is, however, strongly dependent on minimal support value. We set it to 10 in order to eliminate uninteresting patterns. All maximal patterns found for keyword 'DO' you can see below. A trivial pattern with 'do' token was omitted.

```
key(A), pers(A,hamlet), hasWord(A,B), isWord(A,B,I)
key(A), pers(A,hamlet), hasWord(A,B), isWord(A,B,and)
key(A), pers(A,hamlet), hasWord(A,B), isWord(A,B,in)
key(A), pers(A,hamlet), hasWord(A,B), isWord(A,B,it)
key(A), pers(A,hamlet), hasWord(A,B), isWord(A,B,me)
key(A), pers(A,hamlet), hasWord(A,B), isWord(A,B,my)
key(A), pers(A,hamlet), hasWord(A,B), isWord(A,B,not)
key(A), pers(A,polonius), hasWord(A,B), isWord(A,B,you)
key(A), hasWord(A,B), isWord(A,B,your)
key(A), hasWord(A,B), isWord(A,B,what)
key(A), hasWord(A,B), isWord(A,B,we)
key(A), hasWord(A,B), isWord(A,B,think)
key(A), hasWord(A,B), isWord(A,B,they)
key(A), hasWord(A,B), isWord(A,B,them)
key(A), hasWord(A,B), isWord(A,B,lord)
key(A), hasWord(A,B), isWord(A,B,know)
key(A), hasWord(A,B), isWord(A,B,is)
key(A), hasWord(A,B), isWord(A,B,his)
key(A), hasWord(A,B), isWord(A,B,but)
key(A), hasWord(A,B), isWord(A,B,as)
key(A), hasWord(A,B), hasWord(A,C), before(A,B,C),
        isWord(A,B,Do), isWord(A,C,you)
key(A), hasWord(A,B), hasWord(A,C), before(A,B,C),
        isWord(A,B,And), isWord(A,C,do)
key(A), hasWord(A,B), hasWord(A,C), before(A,B,C),
        isWord(A,B,Do), isWord(A,C,not)
```

Among frequent patterns you can find some interesting ones. For example the last one. It says that there are a lot of negations in Hamlet. That can be one of the reasons of frequent occurring 'do' in Hamlet. Another noticeable thing is that ’do’ is pronounced more often by Hamlet than other persons. But, it can be caused by preponderance of Hamlet’s utterances. You can also notice that there are several other keywords among frequent patterns. It is denoted by co-occurrence these keywords with ’do’. In a similar way you can analyze the other keywords.

5 Conclusion

We described classical methods of text keyness representation and introduced a novel way based on relational learning. This method represents keyness as a set of frequent patterns in first-order logic. Hoping that this method can express keyness better then previous methods, we tried to analyze keywords in Hamlet found by Scott. It was shown on an example of ’do’ keyword that relational learning can find more comprehensive information, however, found patterns was not still comprehensive enough.

In our future work we want to extend domain knowledge by other predicates and try to find some other useful information. We will also be interested in so called clouds of patterns. By a special measure we will try to identify near frequent patterns. This clouds may give us a good indication of pattern significance.

Acknowledgments

This work has been partly supported by the Czech Science Foundation under the project 407/07/0679.

References

1. Scott, M.: Key words and key sections: Exploring shakespeare. TALC, Paris (2006).
2. Scott, M., Tribble, C.: Textual Patterns: Key words and corpus analysis in language education. Philadelphia: John Benjamins (2006).
3. Kintsch, W., Van Dijk, T.: Toward a model of text comprehension and production. Psychological Review (1978).
4. Hoey, M.: Patterns of Lexis in Text. Oxford University Press (1991).
5. Phillips, M.: Lexical Structure of Text: Discourse Analysis Monograph 12. University of Birmingham Printing Section (1989).
6. Scott, M.: Lexical analysis software for the PC. Oxford University Press (1996).
7. Popelínský, L.: Keyness and relational learning. In: Keyness in Text, University of Siena (2007).
8. Cussens, J., Džerovski, S.: Learning language in logic. Lecture notes in computer science. Lecture notes in artificial intelligence. Berlin : Springer (2000).
9. Daelemans, W., van den Bosch, A.: Memory-based language processing. Cambridge University Press (2005).
10. Blaťák, J., Popelínský, L., Nepil, M.: RAP: Framework for mining frequent datalog patterns. Proceedings of the first KDID workshop at ECML/PKDD 2002 (2002).

MetaTrans
Multilingual Meta-Translator

Jan Pomikálek

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

Abstract. This paper presents MetaTrans, a meta-search engine for online dictionaries. With this software, users are able to find translations in a number of online dictionaries simultaneously. The MetaTrans features a web interface which is easy to use. The modular design of the tool enables adding support for more online dictionaries with minimal effort. MetaTrans also utilizes information from text corpora, WordNets and a morphological analyzer.

1 Introduction

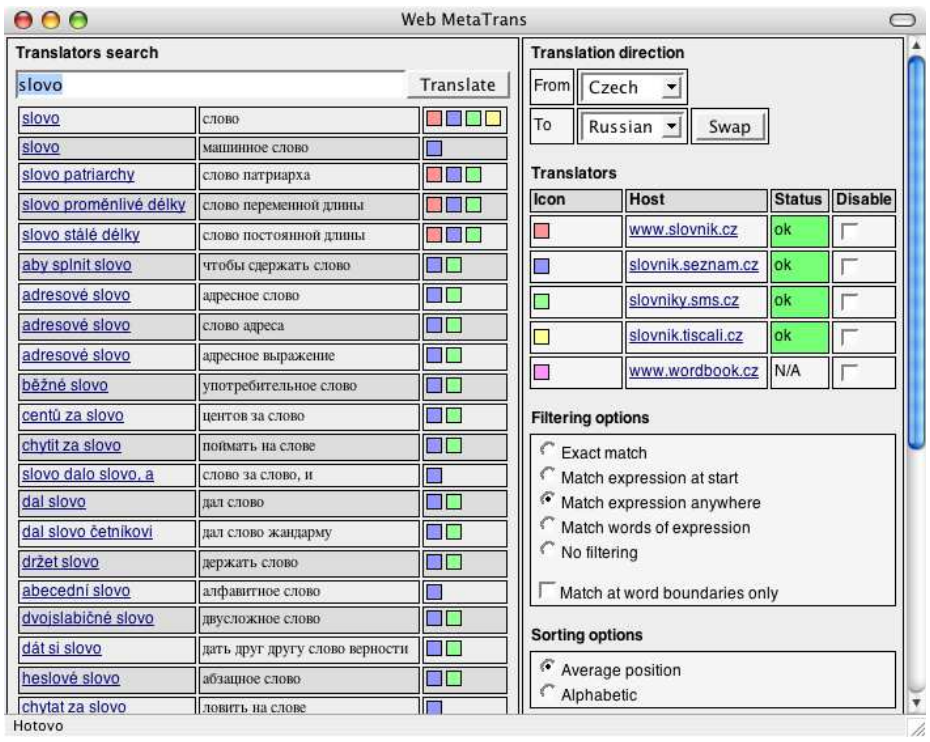
There are many freely available online dictionaries on the web which contain large vocabularies for many language pairs. While common terms are often well covered by each of these dictionaries, the coverage of specialized terms and phrases differs from dictionary to dictionary. An obvious step for a user who did not find a translation of an unknown term in their dictionary is to look into another one. This may mean that the term needs to be searched for multiple times in many different dictionaries until a satisfying translation is found. This can be a tedious and time-consuming process.

We present a system which acts as a meta-translator for a virtually unlimited number of online dictionaries. If a user looks for a translation of term *X* from language *A* into language *B*, the system operates as follows. First, a list of online dictionaries is determined which support translating from *A* to *B*. Then each of these dictionaries is searched for a translation of *X*. The results are merged and returned to the user.

Five online dictionaries are currently supported by the MetaTrans. However, as long as the design of the system is fully modular, it is very easy to add support for more online dictionaries.

The MetaTrans back-end is freely available on CPAN[1]. The MetaTrans web application can be accessed at `http://metatrans.fi.muni.cz/` with some restrictions when used on computers outside Masaryk Universite. These restrictions are specified later in this paper.

[1] `http://search.cpan.org/author/JANPOM/MetaTrans-1.04/bin/metatrans`



Translators search

slovo

slovo	слово	🟥🟦🟩🟨
slovo	машинное слово	🟦
slovo patriarchy	слово патриарха	🟥🟦🟩
slovo proměnlivé délky	слово переменной длины	🟥🟦🟩
slovo stálé délky	слово постоянной длины	🟥🟦🟩
aby splnit slovo	чтобы сдержать слово	🟦🟩
adresové slovo	адресное слово	🟦🟩
adresové slovo	слово адреса	🟦🟩
adresové slovo	адресное выражение	🟦🟩
běžné slovo	употребительное слово	🟦🟩
centů za slovo	центов за слово	🟦🟩
chytit za slovo	поймать на слове	🟦🟩
slovo dalo slovo, a	слово за слово, и	🟦
dal slovo	дал слово	🟦🟩
dal slovo četníkovi	дал слово жандарму	🟦🟩
držet slovo	держать слово	🟦🟩
abecední slovo	алфавитное слово	🟦
dvojslabičné slovo	двусложное слово	🟦🟩
dát si slovo	дать друг другу слово верности	🟦🟩
heslové slovo	абзацное слово	🟦🟩
chytat za slovo	ловить на слове	🟦

Translation direction

From: Czech
To: Russian

Translators

Icon	Host	Status	Disable
🟥	www.slovnik.cz	ok	☐
🟦	slovnik.seznam.cz	ok	☐
🟩	slovniky.sms.cz	ok	☐
🟨	slovnik.tiscali.cz	ok	☐
🟪	www.wordbook.cz	N/A	☐

Filtering options

- ○ Exact match
- ○ Match expression at start
- [✓ ● Match expression anywhere]
- ○ Match words of expression
- ○ No filtering

☐ Match at word boundaries only

Sorting options

- [✓ ● Average position]
- ○ Alphabetic

Fig. 1. MetaTrans main window

2 MetaTrans

MetaTrans is written in pure Perl using object oriented programming. For each online dictionary, a simple object class needs to be created which implements two methods – create_request and process_response.

The **create_request** method accepts three parameters – the term to be translated, the source language and the destination language. It returns a HTTP request which the given online dictionary will respond to with the appropriate list of translations. For example, to translate the word *pes* from Czech to English using the `slovnik.seznam.cz` online dictionary, the following HTTP GET request is required: `http://slovnik.seznam.cz/search.py?wd=pes&lg=cz_en`.

The **process_response** method also accepts three parameters – the response from the online dictionary, the source language and the destination language. The response is returned as an HTML page. The method is responsible for extracting the translations from the HTML code. This can be done either by using regular expressions or using an HTML parser. A relevant part of the response to the request specified above is shown in Fig. 2 The process_response method is responsible for converting it into the list of pairs which is shown in Fig. 3

Additional information is sometimes provided with a translation which specifies the senses of the terms. For the sake of consistency, this information should always be enclosed in parentheses. The process_response method is responsible for the appropriate conversion if necessary. In the previous example, the translations are already in the correct format. However, if we had something like *pes – lovecký* instead of *pes (lovecký)* we would need to convert it.

Apart from the two above mentioned methods, the object class also has to contain a constructor which sets the meta-data of the given online dictionary, such as name and list of supported language pairs.

```
<li> <strong> <a href="search.py?lg=cz_en&wd=pes">pes</a></strong> &nbsp;-&nbsp;
  <a href="search.py?lg=en_cz&wd=tyke">tyke</a> </li>
<li> <strong> <a href="search.py?lg=cz_en&wd=pes">pes</a></strong> &nbsp;-&nbsp;
  <a href="search.py?lg=en_cz&wd=pooch">pooch</a> </li>
<li> <strong> <a href="search.py?lg=cz_en&wd=pes%20%28loveck%C3%BD%29">pes
  (lovecký)</a></strong> &nbsp;-&nbsp; <a href="search.py?lg=en_cz&wd=hound">hound</a>
  <a href="./sound/A/A22051.WAV" title="Přehrát zvuk"><img src="http://1.im.cz/sl/repro.gif"
  width="13" height="13" alt="" class="repro" /></a> </li>
<li> <strong> <a href="search.py?lg=cz_en&wd=pes%20%28t%C3%A9%C5%BE%20p%C5%99en.%29">pes
  (též přen.)</a></strong> &nbsp;-&nbsp; <a href="search.py?lg=en_cz&wd=dog">dog</a>
  <a href="./sound/A/A1331.WAV" title="Přehrát zvuk"><img src="http://1.im.cz/sl/repro.gif"
  width="13" height="13" alt="" class="repro" /></a> </li>
```

Fig. 2. Response sample for `slovnik.seznam.cz`

With the object classes for the online dictionaries, it is already straightforward to search for translations. For a given language pair (A, B) and the input term X, each dictionary is queried for the list of translations of X. In order to speed up the process a new thread is spawned for each dictionary so that all of them can be queried at the same time. When all the responses are retrieved, the results are merged. If the same translation is obtained from multiple sources, it is only displayed once and it is associated with the list of dictionary icons, in which it was found. This is useful information for the user. The more sources of the translation the more likely it is that the translation is correct. If on the other hand, the translation is only found in a single dictionary, it may be that the translation is incorrect or inappropriate.

```
pes - tyke
pes - pooch
pes (lovecký) - hound
pes (též přen.) - dog
```

Fig. 3. The result of processing the response sample

2.1 Sorting

MetaTrans supports two types of sorting the retrieved translations – alphabetically and by average position in the source dictionaries. The alphabetical sorting uses the relevance of the left side to the searched term as the primary criterion. With the left side we refer to the term in the source language. The relevance is determined using the following three rules:

1. The terms which are identical with the searched term are the most relevant.
2. The terms which contain the searched term as a substring are more relevant than the ones which do not contain it (*doggie* is more relevant to *dog* than *hound* is).
3. The more words in the term the lower relevance (*big dog* is more relevant to *dog* than *big bad dog* is).

The terms within the groups with the same relevance are sorted alphabetically, primarily by the left side, secondarily by the right side.

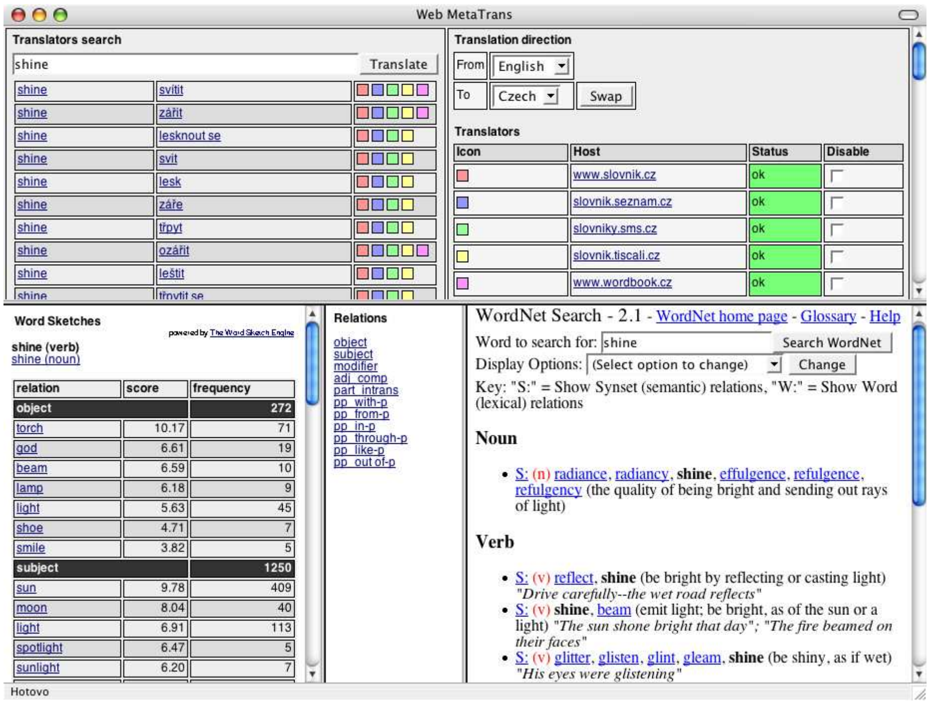
The next supported sorting is by average position and this is the default one. This is based on the assumption that the source dictionaries return the most usual or the most appropriate translations first (at the top). This is the most practical ordering for most users. The sorting by average position attempts to maintain this ordering. For each of the retrieved translations T, and for each source dictionary D, the position (rank) of the T is found within the translations returned by D. If the translation is not found in the list, the position behind the last translation returned by D is used. The positions are then averaged across the dictionaries used. The translations are sorted by the average position.

2.2 Filtering

Several filtering options are available in MetaTrans. The names of the options are mostly self-explanatory. We will therefore only demonstrate the effect of the most important ones on one example. Let us suppose that for a query *dog*, the translation of the following terms were found: *dog*, *doggie*, *dog bite*, *bad dog*, *bad doggie*, *hound*. The results of the filtering are presented in Table 1.

Table 1. The effects of filtering options for the input term *dog*

option	match at word boundaries	
	yes	no
exact match	dog	dog
match expression at start	dog bite	dog bite, doggie
match expression anywhere	dog, dog bite, bad dog	dog, doggie, dog bite, bad dog, bad doggie



shine

shine	svítit	🟥🟦🟩🟨🟪
shine	zářit	🟥🟦🟩🟨🟪
shine	lesknout se	🟥🟦🟩🟨
shine	svit	🟥🟦🟩🟨
shine	lesk	🟥🟦🟩🟨
shine	záře	🟥🟦🟩🟨
shine	třpyt	🟥🟦🟩🟨
shine	ozářit	🟥🟦🟩🟨🟪
shine	leštit	🟥🟦🟩🟨
shine	třpytit se	🟥🟦🟩🟨

Icon	Host	Status	Disable
🟥	www.slovnik.cz	ok	☐
🟦	slovnik.seznam.cz	ok	☐
🟩	slovniky.sms.cz	ok	☐
🟨	slovnik.tiscali.cz	ok	☐
🟪	www.wordbook.cz	ok	☐

Word Sketches

powered by The Word Sketch Engine

shine (verb)
shine (noun)

relation	score	frequency
object		**272**
torch	10.17	71
god	6.61	19
beam	6.59	10
lamp	6.18	9
light	5.63	45
shoe	4.71	7
smile	3.82	5
subject		**1250**
sun	9.78	409
moon	8.04	40
light	6.91	113
spotlight	6.47	5
sunlight	6.20	7

Relations

object
subject
modifier
adj_comp
part_intrans
pp_with-p
pp_from-p
pp_in-p
pp_through-p
pp_like-p
pp_out of-p

WordNet Search - 2.1 - WordNet home page - Glossary - Help

Word to search for: shine [Search WordNet]

Display Options: (Select option to change) [Change]

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Noun

- S: (n) radiance, radiancy, **shine**, effulgence, refulgence, refulgency (the quality of being bright and sending out rays of light)

Verb

- S: (v) reflect, **shine** (be bright by reflecting or casting light) *"Drive carefully--the wet road reflects"*
- S: (v) **shine**, beam (emit light; be bright, as of the sun or a light) *"The sun shone bright that day"; "The fire beamed on their faces"*
- S: (v) glitter, glisten, glint, gleam, **shine** (be shiny, as if wet) *"His eyes were glistening"*

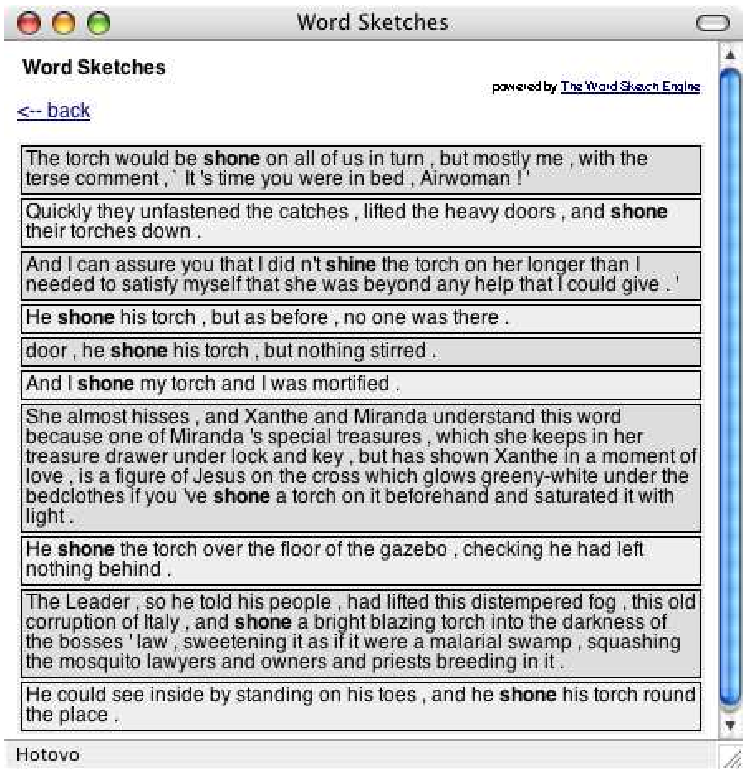
Fig. 4. Word sketches and WordNet information for *shine*

2.3 Additional Resources

MetaTrans can display additional information of various kinds for the retrieved translations. If the user clicks on a term, they are presented with the Word Sketches for the word, with the information from WordNet and with the morphological analysis of the word. Each of these resources is, however, only available for a limited number of languages.

A word sketch [1] is a summary of a word's grammatical and collocational behavior. Word sketches can be produced automatically from large annotated corpora using the Sketch Engine [2]. The collocates in the word sketches table displayed by the MetaTrans can be clicked on in order to display concordances from a text corpus (see Fig. 5). The concordances serve as usage examples of the given word in relation with the collocate.

The word sketches in the MetaTrans are available for Czech, English and French. The British National Corpus [3] is used for English, the Czech National Corpus for Czech. A web derived corpus is used for French. Since the last release of MetaTrans, word sketch tables were developed for a number of additional languages. These will be utilized in the next MetaTrans version. Due to license restrictions, the word sketches information is only available if the MetaTrans is accessed from the Masaryk University network.



The torch would be **shone** on all of us in turn , but mostly me , with the terse comment , ` It 's time you were in bed , Airwoman ! '

Quickly they unfastened the catches , lifted the heavy doors , and **shone** their torches down .

And I can assure you that I did n't **shine** the torch on her longer than I needed to satisfy myself that she was beyond any help that I could give . '

He **shone** his torch , but as before , no one was there .

door , he **shone** his torch , but nothing stirred .

And I **shone** my torch and I was mortified .

She almost hisses , and Xanthe and Miranda understand this word because one of Miranda 's special treasures , which she keeps in her treasure drawer under lock and key , but has shown Xanthe in a moment of love , is a figure of Jesus on the cross which glows greeny-white under the bedclothes if you 've **shone** a torch on it beforehand and saturated it with light .

He **shone** the torch over the floor of the gazebo , checking he had left nothing behind .

The Leader , so he told his people , had lifted this distempered fog , this old corruption of Italy , and **shone** a bright blazing torch into the darkness of the bosses ' law , sweetening it as if it were a malarial swamp , squashing the mosquito lawyers and owners and priests breeding in it .

He could see inside by standing on his toes , and he **shone** his torch round the place .

Fig. 5. Contexts for *shine* in relation with *torch* from the BNC

WordNet [4] is a well-known lexical database which was originally developed for English. In the EuroWordNet project, WordNets were produced for several European languages and linked together. MetaTrans provides web access to the English, Czech and French WordNets.

For Czech words, morphological analysis is available which is provided by the web interface of the morphological analyzer *ajka* [5].

3 Conclusion

We have presented the multilingual meta-translator MetaTrans, a system for parallel searching in multiple online dictionaries. It has a convenient web interface which is easy to use especially for non-technical people. Its parallel processing of the online resources makes MetaTrans reasonably fast. Modular design enables using additional online dictionaries with very little effort. MetaTrans is completely language independent and can be used for translating between any pair of languages, as long as an online dictionary exists for the language pair.

Apart from the data from the online dictionaries, different kinds of language resources are used, such as word sketches, WordNets and a morphological analyzer. This makes MetaTrans unique resource for a wide range of users. Its large vocabulary also makes MetaTrans a valuable dictionary for obtaining specialized terminology.

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. Kilgarriff, A., Tugwell, D.: Sketching words. Lexicography and natural language processing: a festschrift in honour of B. TS Atkins, Euralex (2002) 125–137.
2. Kilgarriff, A., Rychlý, P., Smrž, P., Tugwell, D.: The Sketch Engine. Proceedings of Euralex (2004) 105–116.
3. Aston, G., Burnard, L.: The BNC handbook: Exploring the British National Corpus with SARA. Edinburgh University Press (1998).
4. Miller, G.: WordNet: A Lexical Database for English. Communications of the ACM **38**(11) (1995) p. 39.
5. Sedláček, R., Smrž, P.: A New Czech Morphological Analyser ajka. Proceedings of the TSD (2001) 100–107.

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 2007
Recent Advances in Slavonic Natural Language Processing
First Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2007
Karlova Studánka, Czech Republic, December 14–16, 2007
Proceedings

P. Sojka, A. Horák (Eds.)

Published by Masaryk University, Brno in 2007

Cover design: Petr Sojka

Typesetting: Petr Sojka

Data conversion: Adam Rambousek

Printing: http://librix.eu

First edition, 2007

55-987B-2007 02/58 2/INF

ISBN 978-80-210-4471-5