Linguistic extraction for semantic annotation

Jan Dědek[1] and Peter Vojtáš[2]

[1] Charles University in Prague, Department of Software Engineering
Malostranské nám. 25, 118 00 Prague 1, Czech Republic
jan.dedek@mff.cuni.cz
[2] Academy of Sciences of the Czech Republic, Institute of Computer Science
Pod Vodárenskou věží 2, 182 07 Prague 8, Czech Republic
vojtas@cs.cas.cz

Summary. Bottleneck for semantic web services is lack of semantically annotated information. We deal with linguistic information extraction from Czech texts from the Web for semantic annotation. The method described in the paper exploits existing linguistic tools created originally for a syntactically annotated corpus, Prague Dependency Treebank (PDT 2.0). We propose a system which captures text of web-pages, annotates it linguistically by PDT tools, extracts data and stores the data in an ontology. We focus on the third phase – data extraction – and present methods for learning queries over linguistically annotated data. Our experiments in the domain of reports of traffic accidents enable e.g. summarization of the number of injured people. This serves as a proof of concept of our solution. More experiments, for different queries and different domain are planned in the future. This will improve third party semantic annotation of web resources.

1 Introduction

For the Web to scale, tomorrow's programs must be able to share and process data even when these programs have been designed totally independently. Web services provide a standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks. Web services are characterized by their great interoperability and extensibility, as well as their machine-processable descriptions thanks to the use of XML. They can be combined in a loosely coupled way in order to achieve complex operations. Programs providing simple services can interact with each other in order to deliver sophisticated added-value services [7].

Still, more work needs to be done before the Web service infrastructure can make this vision come true. Current technology around UDDI, WSDL, and SOAP provide limited support in mechanizing service recognition, service configuration and combination (i.e., realizing complex workflows and business logics with Web services), service comparison and automated negotiation. In a business environment, the vision of flexible and autonomous Web service translates into automatic cooperation between enterprise services. Any enterprise requiring a business interaction with another enterprise can automatically discover and select the appropriate optimal Web services

Fig. 1. Example of the web-page with a report of a fire department

relying on selection policies. Services can be invoked automatically and payment processes can be initiated. Any necessary mediation would be applied based on data and process ontologies and the automatic translation and semantic interoperation. An example would be supply chain relationships where an enterprise manufacturing short-lived goods must frequently seek suppliers as well as buyers dynamically. Instead of employees constantly searching for suppliers and buyers, the Web service infrastructure does it automatically within the defined constraints. Other applications areas for this technology are Enterprise-Application Integration (EAI), eWork, and Knowledge Management [6].

Bottleneck for semantic web services is lack of semantically annotated information. This is especially difficult for Web resources described in natural language, especially for IndoEuropean flexitive type languages like Czech Language. We deal with linguistic information extraction from Czech texts from the Web for semantic annotation.

In this paper we describe initial experiments with information extraction from traffic accident reports of fire departments in several regions of the Czech Republic.

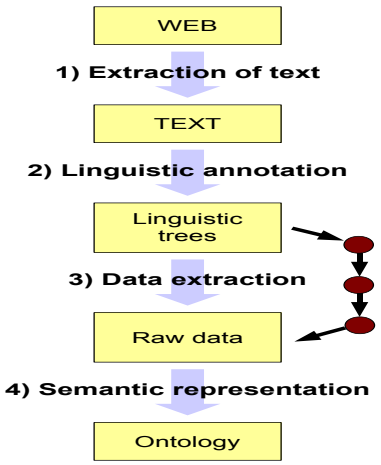


Fig. 2. Schema of the extraction process

These reports are being published on the web[3] of the Ministry of Interior of the Czech Republic. An example of such report can be seen on the Figure 1. We would like to demonstrate the prospects of using linguistic tools from the Prague school of computational linguistic (described in 3). Our experiments are promising, they e.g. enable the summarization of the number of injured people.

Main contributions of this paper are:

1. Experimental chain of tools which captures text of web-pages, annotates it linguistically by PDT tools, extracts data and stores the data in an ontology.
2. In the third phase – data extraction – methods for learning queries over linguistically annotated data.
3. Initial experiments verifying these methods and tools

2 Chain of tools for extraction and annotation

Here we describe our chain of tools for the linguistic extraction of semantic information from text-based web-resources (containing grammatical sentences in a natural language). The chain covers a process that consists of four steps. The Figure 2 describes it. Notice, more detailed structure of the third pahase we focus in this paper.

1. *Extraction of text*
   The linguistic annotating tools process plain text only. In this phase we have to extract the text from the structure of a given web-resource. In this first phase we have used RSS feed of the fire department web-page. From this we have obtained URLs of particular articles and we have downloaded them. Finally we

[3] `http://www.mvcr.cz/rss/regionhzs.html`

have extracted the desired text (see highlighted area in the Figure 1) by means of a regular expression. This text is an input for the second phase.

2. *Linguistic annotation*
   In this phase the linguistic annotators process the extracted text and produce corresponding set of dependency trees representing the deep syntactic structure of individual sentences. We have used the linguistic tools described in the section 3 for this task. Out put of this phase are tectogrammatical trees (for example see Figure 3) of sentences in document under investigation.
3. *Data extraction*
   We use the structure of tectogrammatical (i.e. deep syntactic) dependency trees to extract relevant data. Refinement of this step is the main focus of this paper, see section 4 for more details.
4. *Semantic representation*
   This phase consists of quite simple data transformation or conversion to the desired ontology format. But it is quite important to choose suitable ontology that will properly represent semantics of the data. Output are two fold. An ontology with instances. Annotation of a web resource (e.g. using API to an RDFa editor of html pages).

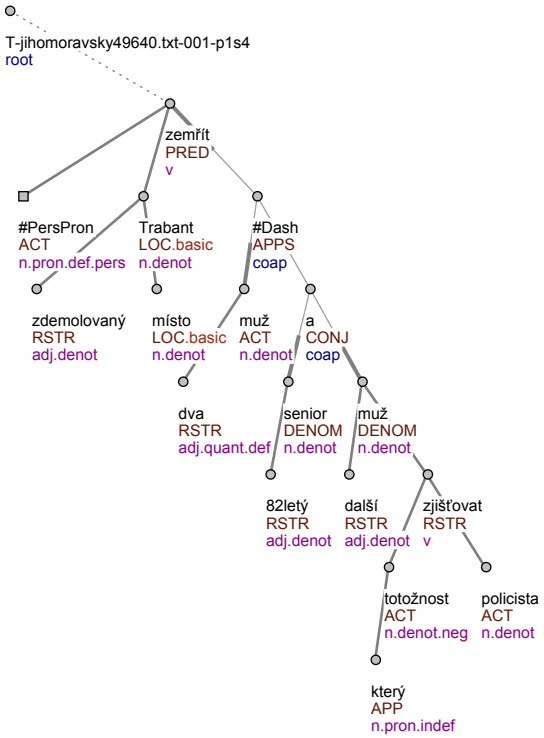


Fig. 3. Example of a tectogrammatical tree

3 PDT linguistic tools for automatic linguistic annotation of texts

In this section we will describe the linguistic tools that we have used to produce linguistic annotation of texts. These tools are being developed in the Institute of Formal and Applied Linguistics[4] in Prague, Czech Republic. They are publicly available – they have been published on a CD-ROM under the title PDT 2.0 [2] (first five tools) and in [3] (Tectogrammatical analysis). These tools are used as a processing chain and at the end of the chain they produce tectogrammatical [4] dependency trees. The Table 1 shows some details about these tools.

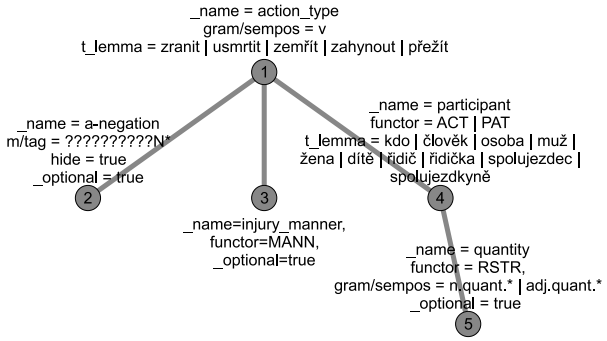
1. **Segmentation and tokenization** consists of tokenization (dividing the input text into words and punctuation) and segmentation (dividing a sequences of tokens into sentences).
2. **Morphological analysis** assigns all possible lemmas and morphological tags to particular word forms (word occurrences) in the text.
3. **Morphological tagging** consists in selecting a single pair lemma-tag from all possible alternatives assigned by the morphological analyzer.
4. **Collins' parser – Czech adaptation** [1]
   Unlike the usual approaches to the description of English syntax, the Czech syntactic descriptions are dependency-based, which means, that every edge of a syntactic tree captures the relation of dependency between a governor and its dependent node. Collins' parser gives the most probable parse of a given input sentence.
5. **Analytical function assignment** assigns a description (*analytical function* – in linguistic sense) to every edge in the syntactic (dependency) tree.
6. **Tectogrammatical analysis** produces linguistic annotation at the tectogrammatical level, sometimes called "layer of deep syntax". Such a tree can be seen on the Figure 3. Annotation of a sentence at this layer is closer to meaning of the sentence than its syntactic annotation and thus information captured at the tectogrammatical layer is crucial for machine understanding of a natural language [3].

Table 1. Linguistic tools for machine annotation

Name of the tool	Results (proclaimed by authors)
Segmentation and tokenization	precision(p): 98,0%, recall(r): 91,4%
Morphological analysis	2,5% unrecognized words
Morphological tagging	93,0% of tags assigned correctly
Collins' parser (Czech adapt.)	precision: 81,6%
Analytical function assignment	precision: 92%
Tectogrammatical analysis [3]	dependencies p: 90,2%, r: 87,9%
	f-tags p: 86,5%, r: 84,3%

[4] `http://ufal.mff.cuni.cz`

4 The linguistic extraction - learning a query



Transcript:

zranit to injure	usmrtit to kill	zemřít to die	zahynout to wane	přežít to survive

kdo somebody	člověk (hu)man	osoba person	muž man	žena woman	dítě child

řidič driver	řidička woman driver	spolujezdec passenger	spolujezdkyně woman passenger

Fig. 4. Netgraph query – extract rule.

Extraction of information in this phase of our research and development is based on specific queries. Here for example, to get from web resources number of injured people in traffic accidents based on concrete traffic accidents reports (in certain time and region - but these are ”easy attributes”). Such an informal query will be translated, in order to be applied to the results of second phase of our process, namely to tectogrammatical trees of traffic accidents reports.

Our linguistic extraction method is based on extraction rules. These rules correspond to query requests of Netgraph application. The Netgraph application [5] is a linguistic tool used for searching through a syntactically annotated corpus of a natural language. It was originally developed for searching the analytical and tectogrammatical levels of the Prague Dependency Treebank, a richly syntactically annotated corpus of Czech [2]. Netgraph queries are written in a special query language. An example of such Netgraph query can be found in the Figure 4. The Netgraph is a general tool for searching trees, it is not limited only to the trees in the PDT format. In our application we use it for searching the tectogrammatical trees provided by a set of language processing tools described in the previous chapter. The tectogrammatical trees have a very convenient property of containing just the type of information we need for our purpose, namely the information about inner participants of verbs - actor, patient, addressee etc.

```
<injured_result>
      <action type="zranit">
            <sentece>
                  Při požáru byla jedna osoba lehce zraněna -- jednalo se
                  o majitele domu, který si vykloubil rameno.
            </sentece>
            <sentece_id>T-vysocina63466.txt-001-p1s4</sentece_id>
            <negation>false</negation>
            <manner>lehký</manner>
            <participant type="osoba">
                  <quantity>1</quantity>
                  <full_string>jedna osoba</full_string>
            </participant>
      </action>
      <action type="zemřít">
            <sentece>
                  Ve zdemolovaném trabantu na místě zemřeli dva muži -- 82letý
                  senior a další muž, jehož totožnost zjišťují policisté.
            </sentece>
            <sentece_id>T-jihomoravsky49640.txt-001-p1s4</sentece_id>
            <negation>false</negation>
            <participant type="muž">
                  <quantity>2</quantity>
                  <full_string>dva muži</full_string>
            </participant>
      </action>
      <action type="zranit">
            <sentece>Čtyřiatřicetiletý řidič nebyl zraněn.</sentece>
            <sentece_id>T-jihomoravsky49736.txt-001-p4s3</sentece_id>
            <negation>true</negation>
            <participant type="řidič">
                  <full_string>Čtyřiatřicetiletý řidič</full_string>
            </participant>
      </action>
</injured_result>
```

Fig. 5. Example of the result of the extraction procedure.

4.1 Extraction method

The extraction works as follows: the extraction rule is in the first step evaluated by searching through a set of syntactic trees. Matching trees are returned and the desired information is taken from particular tree nodes.

Let us explain it in more detail by using the example of extraction rule from the Figure 4. This rule consists of five nodes. Each node of the rule will match some node in each matching tree. So we can investigate the relevant information by reading values of tags of matching nodes. We can find out the number (node number 5) and kind (4) of people, which were or were not (2) killed or injured (1) by an accident that is presented in the given sentence. And we can also identify the manner of injury in the node number 3.

We have evaluated the extraction rule shown in the Figure 4 by using the set of 800 texts of news of several Czech fire departments. There were about 470 sentences matching the rule and we found about 200 numeric values contained in the node number 5. This extraction rule (from the Figure 4) is a result of a learning procedure described in the section 4.2.

Small part of the result of the extraction is shown in the Figure 5. This result contains three pieces of information extracted from three articles.

Each piece of information is closed in the `<action>` element and each deals with some kind of action that happened during some accident.

The attribute `type` specifies the type of the action. So in the first and in the third case there was somebody injured (*zranit* means to injure in Czech) and in the second case somebody died (*zemřít* means to die in Czech).

The element `<negation>` holds the information about negation of the clause. So we can see that the participant of the third action was **not** injured.

The element `<participant>` contains information about the participants of the action. The attribute `type` specifies the type of the participants and the element `<quantity>` holds the number of the participants. So in the first action only a single person (*osoba*) was injured. In the second action two men (*muž*) died and in the third action a driver (*řidič*) was not injured.

4.2 Query learning procedure

So far the process of building up the extraction rules is heavily dependent on skills and experience of a human designer. Fulfillment of this process is quite creative task. But we will try to pick it up as precisely as possible. We assume that a formal description of this process can help us in two ways. First – we can develop tools that will assist the designer of the extraction rules. Second – we can work on the automatization of the process. This process consists of two parts:

Learning the Netgraph query

The procedure of learning the Netgraph query is demonstrated in the Figure 6. One obvious preposition of this learning procedure is that we have a collection of learning texts.

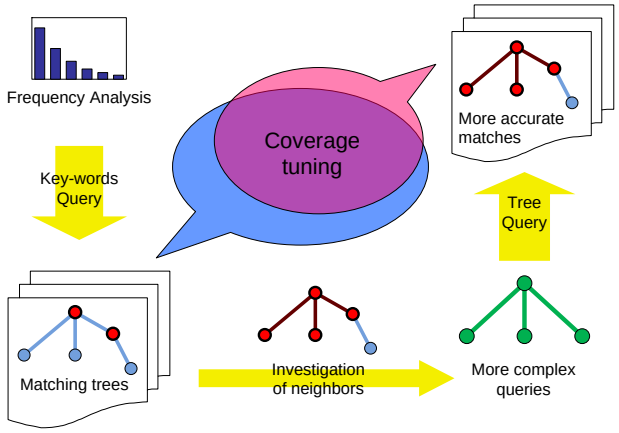


Fig. 6. Schema of the query learning procedure

The procedure starts with frequency analysis of words (their lemmas) occurring in these texts. Especially frequency analysis of verbs is very useful — meaning of a clause is usually strongly dependent on the meaning of corresponding verb.

Frequency analysis helps the designer to choose some representative words (**key-words**) that will be further used for searching the learning text collection. Ideal choice of key-words would cover a majority of sentences that express the information we are looking for and it should cover minimal number of the not-intended sentences (maximization of relevance). An initial choice need not be always sufficient and the process could iterate.

Next step of the procedure consists in **investigating trees** of sentences covered by key-words. System responds with a set of **matching trees**. The designer examines corresponding syntactic trees — looks for the position of key-words and their matching **neighbors** in the trees.

After that the designer can formulate an initial (Netgraph) **tree query** and he or she can compare result of the Netgraph query with the coverage of key-words. Based on this he or she can reformulate the query and gradually **tune** the query and the **query coverage**.

There are two goals of the query tuning. The first goal is maximization of the relevance of the query. The second goal is to involve all important tree-nodes to the query. This second goal is important because the **complexity of the query** (number of involved nodes) makes it possible to extract more complex information. For example see the query on the Figure 4 — each node of it keeps different kind of information.

Semantic interpretation of the query

After the designer have successfully formulated the Netgraph query he or she have to supply semantic interpretation of the query. This interpretation expresses how to transform matching nodes of the query (and the available linguistic information connected with the nodes) to the output data. The complexity of the transformation varies form simple (e.g. putting value of some linguistic attribute of the node to the output) to complex. For example a translation of a numeral to a number can be seen in the Figure 5 (element <quantity>). This is a candidate for our task to select number of killed and/or injured people in traffic accidents. In an inductive procedure (as an another ILP task) we have to learn rules which try to interpret results of extraction procedure in the sense of our task. One example of such rule, can be red as follows: if <negation> has value true, then number of injured people is 0 (e.g. nobody was injured). Another rule can from <negation>false</negation> and <quantity>2</quantity> deduce that number of injured people is two.

Our experiments have shown that the whole chain works and linguistic extraction and semantic annotation are realizable. Nevertheless, it is still a long way to go, especially in automating our process and improving learning on several steps of our procedure.

5 Conclusion

We have presented a proposal of and experiments with a system for linguistic extraction and semantic annotation of information from Czech text on Web pages. Our system relies on linguistic annotation tools from PDT [2] and the tree querying tool Netgraph [5]. Our contributions are an experimental chain of tools which captures text of web-pages, annotates it linguistically by PDT tools, extracts data and stores the data in an ontology. Especially in the third phase – data extraction – we have presented methods for learning queries over linguistically annotated data. Our initial experiments verified these methods and tools. In the near future we would like to extend this method by domain oriented lexical net and semiautomatic search for interesting extraction rules, more experiments with different queries and different domain. In a more distant future we plan to include our method in a semantic web service.

Acknowledgment
This work was partially supported by the Ministry of Education of the Czech Republic (grant MSM0021620838) and by Czech projects 1ET100300517 and 1ET100300419.

References

1. Michael Collins, Jan Hajič, Eric Brill, Lance Ramshaw, and Christoph Tillmann. A Statistical Parser of Czech. In *Proceedings of 37th ACL Conference*, pages 505–512, University of Maryland, College Park, USA, 1999.
2. Jan Hajič, Eva Hajičová, Jaroslava Hlaváčová, Václav Klimeš, Jiří Mírovský, Petr Pajas, Jan Štěpánek, Barbora Vidová-Hladká, and Zdeněk Žabokrtský. Prague dependency treebank 2.0 cd-rom. Linguistic Data Consortium LDC2006T01, Philadelphia 2006, 2006.
3. Václav Klimeš. Transformation-based tectogrammatical analysis of czech. In *Proceedings of the 9th International Conference, TSD 2006*, number 4188 in Lecture Notes In Computer Science, pages 135–142. Springer-Verlag Berlin Heidelberg, 2006.
4. Marie Mikulová, Alevtina Bémová, Jan Hajič, Eva Hajičová, Jiří Havelka, Veronika Kolářová, Lucie Kučová, Markéta Lopatková, Petr Pajas, Jarmila Panevová, Magda Razímová, Petr Sgall, Jan Štěpánek, Zdeňka Urešová, Kateřina Veselá, and Zdeněk Žabokrtský. Annotation on the tectogrammatical level in the prague dependency treebank. annotation manual. Technical Report 30, ÚFAL MFF UK, Prague, Czech Rep., 2006.
5. Jiří Mírovský. Netgraph: A tool for searching in prague dependency treebank 2.0. In Jan Hajič and Joakim Nivre, editors, *Proceedings of the Fifth Workshop on Treebanks and Linguistic Theories (TLT)*, number 5, pages 211–222, Prague, Czech rep., 2006.
6. SWSI. Semantic web services initiative.
7. W3C. Web services activity statement, 2008.