Linguistic extraction for semantic annotation

Jan Dědek (contact author)
Department of software engineering
School of Computer Science
Faculty of Mathematics and Physics
Charles University in Prague
Malostranské nám. 25
118 00 Praha 1
Czech Republic
Tel: +420-221-914-264
Fax: +420-221-914-323
Email: jan.dedek@mff.cuni.cz

Peter Vojtáš
Department of software engineering
School of Computer Science
Faculty of Mathematics and Physics
Charles University in Prague
Malostranské nám. 25
118 00 Praha 1
Czech Republic
Tel: +420-221-914-264
Fax: +420-221-914-323
Email: vojtas@cs.cas.cz

Conference title: **SWWS'08**

***Abstract*—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.**

***Index Terms*—semantic web mining; tools, languages and techniques for semantic annotation of web data; linguistic extraction, semantic annotation**

I. 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 [1].

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 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 [2].

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.

Linguistic extraction for semantic annotation of data from Web resources can be seen in three dimensions. The first dimension is the amount of human work associated with the extraction and annotation method. This dimension ranges from human "hand written" through human assisted, semiautomatic (human trained) to fully automatic approach. The second dimension is the variety and complexity of queries (concepts) instances of which should be extracted and annotated. The third dimension of the problem is the independence on application domain. Problem of selectivity of the extraction task — ranging from document classification thorough data region and data record recognition to the extraction of attribute values can be considered as an additional dimension, see e.g. [3].

In this paper we try to proceed in all dimensions in the "hard" direction. Nevertheless we are just in the begining.

A. Motivation

In this paper we describe initial experiments with information extraction from traffic accident reports of fire departments in several regions of the Czech Republic. We would like to demonstrate the prospects of using linguistic tools from the Prague school of computational linguistic (described in III). These experiments are promising, they e.g. enable the summarization of the number of injured people.

The Ministry of Interior of the Czech Republic presents on its Web pages[1] also reports from fire departments of several regions of the Czech Republic. These departments are responsible for rescue and recovery after traffic accidents. These reports are rich in information, e.g. where and when an traffic accident occurred, which units helped, how much time it took them to show up on the place of accident, how many people were injured, killed etc. An example of such report can be seen in the Figure 2.

B. Main contributions

The method described in the paper exploits existing linguistic tools created originally for a syntactically annotated corpus, Prague Dependency Treebank (PDT 2.0) [4].

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 (annotates).
2) In the third phase – data extraction – methods for learning queries over linguistically annotated data.
3) Initial experiments verifying these methods and tools

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] http://www.mvcr.cz/rss/regionhzs.html

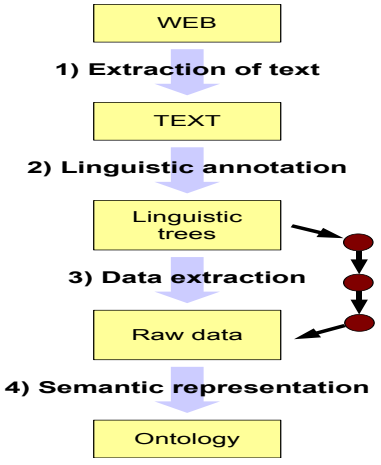


Fig. 1. Schema of the extraction process

Fig. 2. Example of the web-page with a report of a fire department

Nevertheless, there are also reports about fire accidents and also about fire-fighting contests. The task to extract information on the number of injured/killed people in traffic accidents needs deep linguistic analysis.

In an ideal case all these reports would be already semantically annotated, e.g. using RDFa [5]. In this case extraction could be done fully automatically by a software agent. In our case, there is no semantic annotation present in the reports.

II. Chain of tools for extraction and annotation

Our chain of tools for the linguistic extraction and semantic annotation of linguistic data from text-based web-resources (grammatical sentences in a natural language). This process consists of four steps. The Figure 1 describes it.

Notice, more detailed structure of the third process 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 have extracted the desired text (see highlighted area in the Figure 2) by means of a regular expression. This text is an input for the second phase.

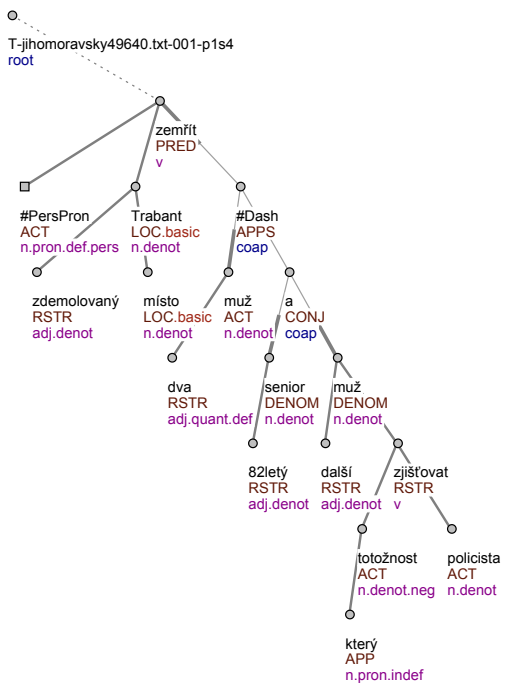


Fig. 3. Example of a tectogrammatical tree

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 III 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 IV 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).

III. 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[2] in Prague, Czech Republic. They are publicly available – they have been published on a CD-ROM under the title PDT 2.0 [4] (first five tools) and in [6] (Tectogrammatical analysis). These tools are used as a processing chain and at the end of the chain they produce tectogrammatical [7] dependency trees. The Table I shows some details about these tools.

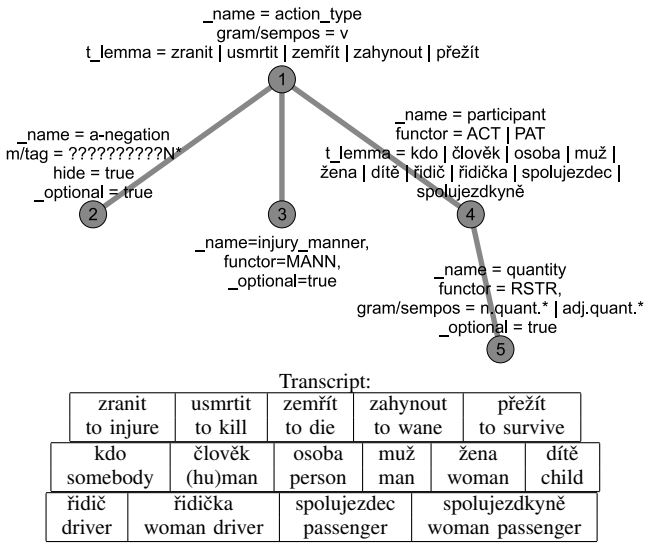
[2]http://ufal.mff.cuni.cz

TABLE I
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 [6]	dependencies p: 90,2%, r: 87,9% f-tags p: 86,5%, r: 84,3%

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** [8]
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 [6].

Although all tools mentioned above are a part of the processing chain mentioned in the previous chapter, the errors they produce are not multiplied. The numbers listed in the table are measured against a corresponding level of the Prague Dependency Treebank. If, for example, the morphological tagger, which is used as a prerequisite of the analytical parser (Collins' parser), has a precision of 93%, the 7% of incorrectly assigned tags definitely have some influence on the quality of the result of the parser. This influence is nevertheless already reflected in the 81,6% precision of the parser. The results of the parser are measured against the correct trees contained in the



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.

```
<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.

PDT, therefore the precision of the parser is the **total** precision of the combination of the tagger and the parser. On the other hand, the (linguistic) analytical function assignment and the tectogrammatical analysis are both more or less dependent on the results of the previous phases and their precision has to be taken as a precision obtained on (already slightly imprecise) results of the analytical parser.

These facts are important especially wrt. possible extension of the system in the future. It might be an interesting research topic to compare the results of the information extraction obtained through the exploitation of the full processing chain versus the results of the analytical parser only - the complications caused by the differences between the analytical level of representation and the desired structure of extracted information might be compensated by decreasing the amount of imprecision present in the system.

IV. The linguistic extraction - learning a query

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.

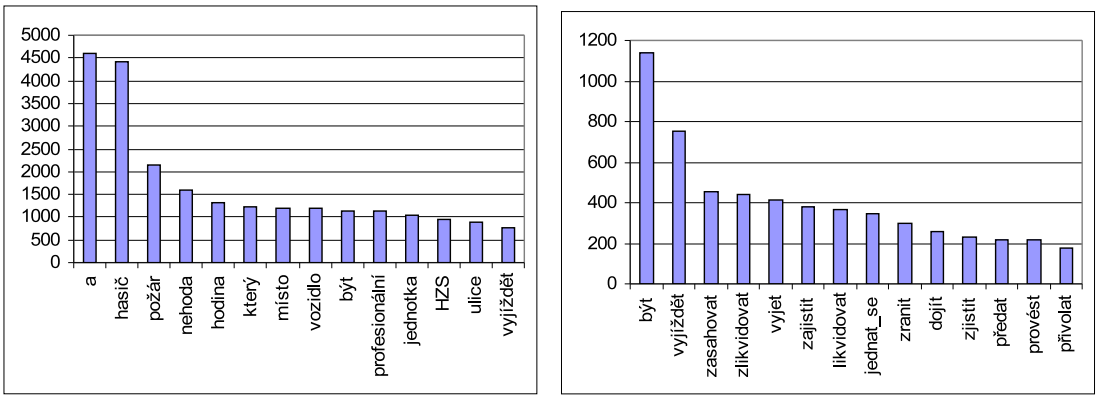
Our linguistic extraction method is based on extraction rules. These rules correspond to query requests of Netgraph application. The Netgraph application [9] is a linguistic tool used for searching through a syntactically annotated corpus of a natural language. It was originally developed for searching the analytical and tectogrammatical levels of the Prague Dependency Treebank, a richly syntactically annotated corpus of Czech [4]. 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 a more detail 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.

The tectogrammatical (deep syntactic) level of representation is more suitable for our purpose than the analytical (surface syntactic) level of representation of the structure of each sentence. The inner participants (actor, patient, addressee etc.) provide much more reliable information about the actual meaning of the sentence than the syntactic roles (subject, object etc.). The roles of a subject or object are misleading especially in the case of passive sentences, where usually the subject of the sentence corresponds to the patient or addressee while the actor is expressed by an object of the passive sentence. In these cases it would be necessary to develop (for the analytical level representation) some kind of algorithm analyzing these cases and providing the assignment of proper roles to individual words, while at the tectogrammatical level we get the desired information directly.

A. 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



Transcript:
a – and, *hasič* – fireman, *požár* – fire, *nehoda* – accident, *hodina* – hour, *který* – which, *místo* – place, *vozidlo* – vehicle, *být* – to be, *profesionální* – professional, *jednotka* – unit, *HZS* – fire department, *ulice* – street, *vyjíždět* – to drive out, *zasahovat* – to intervene, *vyjet* – to rush off, *zajistit* – to ensure, *likvidovat* – to liquidate, *jdnat se* – to deal, *zranit* – to injure, *dojít* – to reach, *zjistit* – to find out, *předat* – to hand over, *provést* – to make, *přivolat* – to call in

Fig. 6. Frequency analysis (most frequented words only), the right figure – restriction on verbs.

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 IV-B.

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.

B. 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:

1) Learning the Netgraph query: The procedure of learning the Netgraph query is demonstrated in the Figure 7. One obvious preposition of this learning procedure is that we have a collection of learning texts.

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. An example of frequency analysis of our data can be seen on the Figure 6.

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

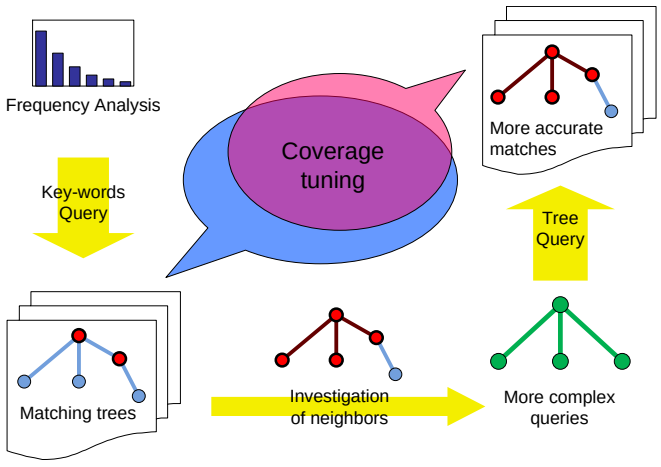


Fig. 7. Schema of the query learning procedure

number of the not-intended sentences (maximization of relevance). An initial choice need not be always sufficient and the process could iterate. Notice that we understand our data as a stream and we are not calculating inverse document frequency of terms, hence we use this simple approximation for first selection. Our first results is that frequency analysis of verbs gives better results. Selecting the threshold above which start the verb be interesting can also iterate. In Figure 6 the verb threshold is set to 180, but an important Czech verb "utrpět" (*utrpět zranění* means sustain injury) shows at threshold about 70.

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.

The above text was personalized, and described a human training of Netgraph query formulation. For semiautomatic procedure we have to have a method to detect failures. This can be done by learning logic programming rules. We plan to use an Inductive Logic Programming ILP method and tool in future. Moreover, our learning procedure has several parameters (e.g. frequency threshold). To tune this parameters is also an iterative task and we have to decide the trade off between relevance of query in one step (smaller threshold means more relevant terms will occur in the query) and efficiency in the next step (bigger number of matching trees makes more difficult to investigate neighbors and formulate more complex queries). This is a topic for future research.

2) 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, tuning our parameters and improving learning on several steps of our procedure.

V. 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 [4] and the tree querying tool Netgraph [9]. 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 (annotates). 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 Czech projects 1ET100300517 and 1ET100300419.

References

[1] W3C, "Web services activity statement," 2008. [Online]. Available: http://www.w3.org/2002/ws/Activity

[2] SWSI, "Semantic web services initiative." [Online]. Available: http://www.swsi.org/

[3] C.-H. Chang, M. Kayed, M. R. Girgis, and K. F. Shaalan, "A survey of web information extraction systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 10, pp. 1411–1428, 2006.

[4] J. Hajič, E. Hajičová, J. Hlaváčová, V. Klimeš, J. Mírovský, P. Pajas, J. Štěpánek, B. Vidová-Hladká, and Z. Žabokrtský, "Prague dependency treebank 2.0 cd-rom," Linguistic Data Consortium LDC2006T01, Philadelphia 2006, Philadelphia, 2006.

[5] W3C, "Rdfa primer," 2007. [Online]. Available: http://www.w3.org/TR/xhtml-rdfa-primer/

[6] V. Klimeš, "Transformation-based tectogrammatical analysis of czech," in *Proceedings of the 9th International Conference, TSD 2006*, ser. Lecture Notes In Computer Science, no. 4188. Springer-Verlag Berlin Heidelberg, 2006, pp. 135–142.

[7] M. Mikulová, A. Bémová, J. Hajič, E. Hajičová, J. Havelka, V. Kolářová, L. Kučová, M. Lopatková, P. Pajas, J. Panevová, M. Razímová, P. Sgall, J. Štěpánek, Z. Urešová, K. Veselá, and Z. Žabokrtský, "Annotation on the tectogrammatical level in the prague dependency treebank. annotation manual," ÚFAL MFF UK, Prague, Czech Rep., Tech. Rep. 30, 2006.

[8] M. Collins, J. Hajič, E. Brill, L. Ramshaw, and C. Tillmann, "A Statistical Parser of Czech," in *Proceedings of 37th ACL Conference*, University of Maryland, College Park, USA, 1999, pp. 505–512.

[9] J. Mírovský, "Netgraph: A tool for searching in prague dependency treebank 2.0," in *Proceedings of the Fifth Workshop on Treebanks and Linguistic Theories (TLT)*, J. Hajič and J. Nivre, Eds., no. 5, Prague, Czech rep., 2006, pp. 211–222.