Semantic Annotation Semantically: Using a Shareable Extraction Ontology and a Reasoner

Jan Dědek
Department of Software Engineering
MFF, Charles University
Prague, Czech Republic
dedek@ksi.mff.cuni.cz

Peter Vojtáš
Department of Software Engineering
MFF, Charles University
Prague, Czech Republic
vojtas@ksi.mff.cuni.cz

***Abstract*—Information extraction (IE) and automated semantic annotation of text are usually done by complex tools. These tools use some kind of a model that represents the actual task and its solution. The model is usually represented as a set of extraction rules (e.g., regular expressions), gazetteer lists, or it is based on some statistical measurements and probability assertions. In the environment of the Semantic Web it is essential that information is shareable and some ontology based IE tools keep the model in so called extraction ontologies. In practice, the extraction ontologies are usually strongly dependent on a particular extraction/annotation tool and cannot be used separately. In this paper, we present an extension of the idea of extraction ontologies. According to the presented concept the extraction ontologies should not be dependent on the particular extraction/annotation tool. In our solution the extraction/annotation process can be done separately by an ordinary reasoner. We also present a proof of concept for the idea: a case study with a linguistically based IE engine that exports its extraction rules to an extraction ontology and we demonstrate how this extraction ontology can be applied to a document by a reasoner. The paper also contains an evaluation experiment with several OWL reasoners.**

Keywords-Extraction Ontology; Reasoning; Information Extraction; Semantic Annotation;

I. Introduction

Information extraction (IE) and automated semantic annotation of text are usually done by complex tools and all these tools use some kind of model that represents the actual task and its solution. The model is usually represented as a set of some kind of extraction rules (e.g., regular expressions), gazetteer lists or it is based on some statistical measurements and probability assertions (classification algorithms like Support Vector Machines (SVM), Maximum Entropy Models, Decision Trees, Hidden Markov Models (HMM), Conditional Random Fields (CRF), etc.)

In the beginning, a model is either created by a human user or it is learned from a training dataset. Then, in the actual extraction/annotation process, the model is used as a configuration or as a parameter of the particular extraction/annotation tool. These models are usually stored in proprietary formats and they are accessible only by the corresponding tool.

In the environment of the Semantic Web it is essential that information is shareable and some ontology based IE tools keep the model in so called extraction ontologies [1]. Extraction ontologies should serve as a wrapper for documents of a narrow domain of interest. When we apply an extraction ontology to a document, the ontology identifies objects and relationships present in the document and it associates them with the corresponding ontology terms and thus wraps the document so that it is understandable in terms of the ontology [1].

In practice the extraction ontologies are usually strongly dependent on a particular extraction/annotation tool and cannot be used separately. The strong dependency of an extraction ontology on the corresponding tool makes it very difficult to share. When an extraction ontology cannot be used outside the tool there is also no need to keep the ontology in a standard ontology format (RDF or OWL).

The only way how to use such extraction ontology is within the corresponding extraction tool. It is not necessary to have the ontology in a "owl or rdf file". In a sense such extraction ontology is just a configuration file. For example in [2] (and also in [1]) the so called extraction ontologies are kept in XML files with a proprietary structure and it is absolutely sufficient, there is no need to treat them differently.

A. Shareable Extraction Ontologies

In this paper, we present an extension of the idea of extraction ontologies. We adopt the point that extraction models are kept in extraction ontologies and we add that the extraction ontologies should not be dependent on the particular extraction/annotation tool. In such case the extraction/annotation process can be done separately by an ordinary reasoner.

In this paper, we present a proof of concept for the idea: a case study with our linguistically based IE engine and an experiment with several OWL reasoners. In the case study (see Section IV) the IE engine exports its extraction rules to the form of an extraction ontology. Third party linguistic tool linguistically annotates an input document and the linguistic

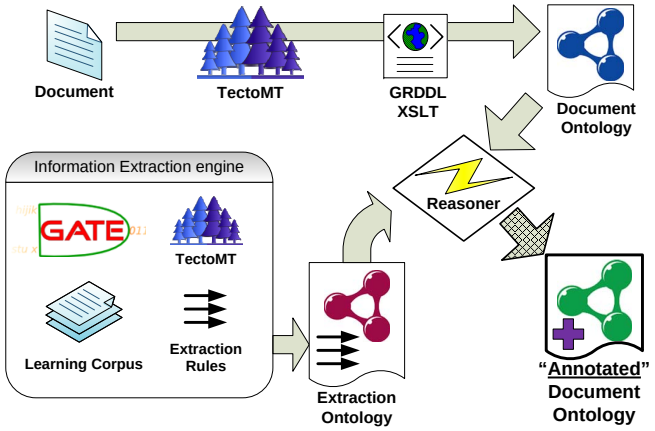

Figure 1. Semantic annotation driven by an extraction ontology and a reasoner – schema of the process.

annotations are translated to so-called document ontology. After that an ordinary OWL reasoner is used to apply the extraction ontology on the document ontology, which has the same effect as a direct application of the extraction rules on the document. The process is depicted in Fig 1 and it will be described in detail in Section IV-B.

Section II presents several closely related works. The main idea of the paper will be described in Section III, its implementation in Section IV and in Section V an experiment with several OWL reasoners and IE datasets will be presented. In Section VI related issues are discussed and Section VII concludes the paper.

II. Related Work

Ontology-based Information Extraction (OBIE) [3] or Ontology-driven Information Extraction [4] has recently emerged as a subfield of information extraction. Furthermore, Web Information Extraction [5] is a closely related discipline. Many extraction and annotation tools can be found in the above mentioned surveys ([3], [5]), many of the tools also use an ontology as the output format, but almost all of them store their extraction models in proprietary formats and the models are accessible only by the corresponding tool.

In the literature we have found only two approaches that use extraction ontologies. The former one was published by D. Embley [1], [6] and the later one – IE system Ex was developped by M. Labský [2]. But in both cases the extraction ontologies are dependent on the particular tool and they are kept in XML files with a proprietary structure.

By contrast authors of [3] (a recent survey of OBIE systems) do not agree with allowing for extraction rules to be a part of an ontology. They use two arguments against that:

1) Extraction rules are known to contain errors (because they are never 100% accurate), and objections can be raised on their inclusion in ontologies in terms of formality and accuracy.
2) It is hard to argue that linguistic extraction rules should be considered a part of an ontology while information extractors based on other IE techniques (such as SVM, HMM, CRF, etc. classifiers used to identify instances of a class when classification is used as the IE technique) should be kept out of it: all IE techniques perform the same task with comparable effectiveness (generally successful but not 100% accurate). But the techniques advocated for the inclusion of linguistic rules in ontologies cannot accommodate such IE techniques.

   The authors then conclude that either all information extractors (that use different IE techniques) should be included in the ontologies or none should be included.

Concerning the first argument, we have to take into account that extraction ontologies are not ordinary ontologies, it should be agreed that they do not contain 100% accurate knowledge. Also the estimated accuracy of the extraction rules can be saved in the extraction ontology and it can then help potential users to decide how much they will trust the extraction ontology.

Concerning the second argument, we agree that in the case of complex classification based models (SVM, HMM, CRF, etc.) serialization of such model to RDF does not make much sense (cf. the next section). But on the other hand we think that there are cases when shareable extraction ontologies can be useful and in the context of Linked Data providing shareable descriptions of information extraction rules may be valuable. It is also possible that new standard ways how to encode such models to an ontology will appear in the future.

III. Semantic Annotation Semantically

The problem of extraction ontologies that are not shareable was pointed out in the introduction (Section I). The cause of the problem is that a particular extraction model can only be used and interpreted by the corresponding extraction tool. If an extraction ontology should be shareable, there has to be a commonly used tool that is able to interpret the extraction model encoded by the extraction ontology. In this paper we present a proof of concept that Semantic Web reasoners can play the role of commonly used tools that can interpret shareable extraction ontologies.

Although it is probably always possible to encode an extraction model using a standard ontology language, only certain way of encoding makes it possible to interpret such model by a standard reasoner in the same way as if the original extraction tool was used. The difference is in semantics. It is not sufficient to encode just the model's data, it is also necessary to encode the semantics of the model. Only then the reasoner is able to interpret the model in the same way as the original tool. And this is where the title of the paper and the present section comes from. If the process of information extraction or semantic annotation should be

performed by an ordinary Semantic Web reasoner then only means of semantic inference are available and the extraction process must be correspondingly semantically described.

In the presented solution the approaching support for Semantic Web Rule Language (SWRL) [7] is exploited. Although SWRL is not yet approved by W3C it is already widely supported by Semantic Web tools including many OWL reasoners. The SWRL support makes it much easier to transfer the semantics of extraction rules used by our IE tool. The case study in Section IV demonstrates the translation of the native extraction rules to SWRL rules that form the core of the extraction ontology.

IV. The Main Idea Illustrated – a Case Study

In this section, realization of the main idea of the paper will be described and illustrated on a case study.

A. Document Ontologies

The main idea of this paper assumes that extraction ontologies will be shareable and they can be applied on a document outside of the original extraction/annotation tool. We further assert that the extraction ontologies can be applied by ordinary reasoners. This assumption implies that both extraction ontologies and documents have to be in a reasoner readable format. In the case of contemporary OWL reasoners there are standard reasoner-readable languages: OWL and RDF in a rich variety of possible serializations (XML, Turtle, N-Triples, etc.) Besides that there exists standard ways like GRDDL or RDFa how to obtain a RDF document from an "ordinary document" (strictly speaking XHTML and XML documents).

We call 'document ontology' an ontology that formally captures content of a document. A document ontology can be for example obtained from the source document by a suitable GRDDL transformation (as in our experiment). A document ontology should contain all relevant data of a document and preferably the document could be reconstructed from the document ontology on demand.

When a reasoner is applying an extraction ontology to a document, it only has "to annotate" the corresponding document ontology, not the document itself. Here "to annotate" means to add new knowledge – new class membership or property assertions. In fact it means just to do the inference tasks prescribed by the extraction ontology on the document ontology.

Obviously when a document can be reconstructed from its document ontology (this is very often true, it is necessary just to save all words and formatting instructions) then also an annotated document can be reconstructed from its annotated document ontology.

B. Implementation

In this section, we will present details about the case study. We have used our IE engine [8] based on deep linguistic parsing and Inductive Logic Programming. It is a complex system implemented with a great help of the GATE system (http://gate.ac.uk/) and it also uses many other third party tools including several linguistic tools and a Prolog system. Installation and making the system operate is not simple. This case study should demonstrate that the extraction rules produced by the system are not dependent on the system in the sense described above.

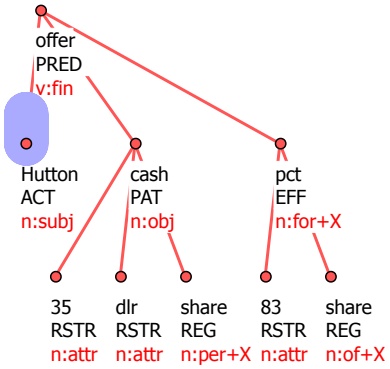

Figure 2. Tectogrammatical tree of the sentence: "Hutton is offering 35 dlrs cash per share for 83 pct of the shares." Nodes roughly correspond with words of a sentence, edges represent linguistic dependencies between nodes and some linguistic features (tectogrammatical lemma, semantic functor and semantic part of speech) are printed under each node.

1) Linguistic Analysis: Our IE engine needs a linguistic preprocessing (deep linguistic parsing) of documents on its input. Deep linguistic parsing brings a very complex structure to the text and the structure serves as a footing for construction and application of extraction rules.

We usually use TectoMT system [9] to do the linguistic preprocessing. TectoMT is a Czech project that contains many linguistic analyzers for different languages including Czech and English. We are using a majority of applicable tools from TectoMT: a tokeniser, a sentence splitter, morphological analyzers (including POS tagger), a syntactic parser and the deep syntactic (tectogrammatical) parser. All the tools are based on the dependency based linguistic theory and formalism of the Prague Dependency Treebank (PDT, http://ufal.mff.cuni.cz/pdt2.0/).

The output linguistic annotations of the TectoMT system are stored (along with the text of the source document) in XML files in so called Prague Markup Language (PML, http://ufal.mff.cuni.cz/jazz/PML/). PML is a very complex language (or XML schema) that is able to express many linguistic elements and features present in text. For the IE engine a tree dependency structure of words in sentences is the most useful one because the edges of the structure guide the extraction rules. An example of such (tectogrammatical) tree structure is in Fig. 2.

In this case study, PML files made from source documents by TectoMT are transformed to RDF document ontology by quite simple GRDDL/XSLT transformation. Such document ontology contains the whole variety of PML in RDF format.

```
[Rule 1] [Pos cover = 23 Neg cover = 6]
mention_root(acquired,A) :-
   'lex.rf'(B,A), t_lemma(B,'Inc'),
   tDependency(C,B), tDependency(C,D),
   formeme(D,'n:in+X'), tDependency(E,C).
[Rule 11] [Pos cover = 25 Neg cover = 6]
mention_root(acquired,A) :-
   'lex.rf'(B,A), t_lemma(B,'Inc'),
   tDependency(C,B), formeme(C,'n:obj'),
   tDependency(C,D), functor(D,'APP').
[Rule 75] [Pos cover = 14 Neg cover = 1]
mention_root(acquired,A) :-
   'lex.rf'(B,A), t_lemma(B,'Inc'),
   functor(B,'APP'), tDependency(C,B),
   number(C,pl).
```

Figure 3. Examples of extraction rules in the native Prolog format.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Ontology [
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
  <!ENTITY pml "http://ufal.mff.cuni.cz/pdt/pml/" >
]>
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
  ontologyIRI="http://czsem.berlios.de/onto ... rules.owl">
  <DLSafeRule>
    <Body>
      <ObjectPropertyAtom>
        <ObjectProperty IRI="&pml;lex.rf" />
        <Variable IRI="urn:swrl#b" />
        <Variable IRI="urn:swrl#a" />
      </ObjectPropertyAtom>
...
      <DataPropertyAtom>
        <DataProperty IRI="&pml;number" />
        <Variable IRI="urn:swrl#c" />
        <Literal>pl</Literal>
      </DataPropertyAtom>
    </Body>
    <Head>
      <DataPropertyAtom>
        <DataProperty IRI="&pml;mention_root" />
        <Literal>acquired</Literal>
        <Variable IRI="urn:swrl#a" />
      </DataPropertyAtom>
    </Head>
  </DLSafeRule>
</Ontology>
```

Figure 4. Rule 75 in the OWL/XML syntax for Rules in OWL 2 [10].

```
@prefix pml: <http://ufal.mff.cuni.cz/pdt/pml/>.
[rule-75:
        ( ?b pml:lex.rf ?a )
        ( ?c pml:tDependency ?b )
        ( ?b pml:functor 'APP' )
        ( ?c pml:number 'pl' )
        ( ?b pml:t_lemma 'Inc' )
     ->
        ( ?a pml:mention_root 'acquired' )
]
```

Figure 5. Rule 75 in the Jena rules syntax.

2) Rule Transformations: Extraction rules produced by the IE engine are natively kept in a Prolog format; examples can be seen in Fig. 3. The engine is capable to export them to the OWL/XML syntax for rules in OWL 2 [10] (see in Fig. 4). Such rules can be parsed by OWL API (http://owlapi.sourceforge.net/) 3.1 and exported to RDF/SWRL, which is very widely supported and hopefully becoming a W3C recommendation. The last rule example can be seen in Fig. 5, it shows a rule in the Jena rules format. Conversion to Jena rules was necessary because it is the only format that Jena can parse, see details about our use of Jena in Section V. The Jena rules were obtained using following transformation process: OWL/XML → RDF/SWRL conversion using OWL API and RDF/SWRL → Jena rules conversion using SweetRules (http://sweetrules.semwebcentral.org/).

The presented rules belong to the group of so called DL-Safe rules [11] so the decidability of OWL reasoning is kept.

3) Schema of the Case Study: A schema of the case study was presented in Fig. 1. The top row of the image illustrates how TectoMT (third party linguistic tool) linguistically annotates an input document and the linguistic annotations are translated to so-called document ontology by a GRDDL/XSLT transformation.

In the bottom of the picture our IE engine learns extraction rules and exports them to an extraction ontology. The reasoner in the middle is used to apply the extraction ontology on the document ontology and it produces the "annotated" document ontology, which was described in Section IV-A.

V. Experiment

In this section, we present an experiment that should serve as a proof of a concept that the proposed idea of independent extraction ontologies is realizable. We have selected several reasoners (namely Jena, HermiT, Pellet and FaCT++) and tested them on two slightly different datasets from two different domains and languages (see Table I). This should at least partially demonstrate the universality of the proposed approach.

In both cases the task is to find all instances (corresponding to words in a document) that should be uncovered by the extraction rules. The extraction rules are saved in single extraction ontology for each dataset. The datasets are divided into individual document ontologies (owl files) corresponding to the individual documents. During the experiment the individual document ontologies are processed separately (one ontology in a step) by a selected reasoner. The total time taken to process all document ontologies of a dataset is the measured result of the reasoner for the dataset.

The actual reasoning tasks are more difficult than a simple retrieval of all facts entailed by the extraction rules. Such simple retrieval task took only a few seconds for the Acquisitions v1.1 dataset (including parsing) in the native Prolog environment that the IE engine uses. There were several more inferences needed in the reasoning tasks because the schema of the input files was a little bit different from the schema used in rules. The mapping of the schemas was captured in another "mapping" ontology that was included in the reasoning. The mapping ontology is a part of the publically available project ontologies.

A. How to Download

All the resources (including source codes of the case study and the experiment, datasets and ontologies) mentioned in this paper are publically available on the project's web site (http://czsem.berlios.de/ (before 2012) or http://czsem.

Table I
DESCRIPTION OF DATASETS THAT WERE USED.

dataset	domain	language	number of files	dataset size (MB)	number of rules
czech_fireman	accidents	Czech	50	16	2
acquisitions	finance	English	600	126	113

sourceforge.net/) and detailed information can be found there.

B. Datasets

In the experiment we used two slightly different datasets from two different domains and languages. Table I summarizes some basic information about them.

1) Czech Fireman: The fist dataset is called 'czech_fireman'. This dataset was created by ourselves during the development of our IE engine. It is a collection of 50 Czech texts that are reporting on some accidents (car accidents and other actions of fire rescue services). These reports come from the web of Fire rescue service of Czech Republic. The corpus is structured such that each document represents one event (accident) and several attributes of the accident are marked in text. For the experiment we selected the 'damage' task – to find an amount (in CZK - Czech Crowns) of summarized damage arisen during a reported accident.

2) Acquisitions v1.1: The second dataset is called "Corporate Acquisition Events". More precisely we used the *Acquisitions v1.1* version[1] of the corpus. This is a collection of 600 news articles describing acquisition events taken from the Reuters dataset. News articles are tagged to identify fields related to acquisition events. These fields include 'purchaser' , 'acquired', and 'seller' companies along with their abbreviated names ('purchabr', 'acqabr' and 'sellerabr'). Some news articles also mention the field 'deal amount'. For the experiment we selected only the 'acquired' task.

C. Reasoners

In the experiment we used four OWL reasoners:

Jena (http://jena.sourceforge.net),
HermiT (http://hermit-reasoner.com),
Pellet (http://clarkparsia.com/pellet),
FaCT++ (http://code.google.com/p/factplusplus) .

We measured the time they spent on processing a particular dataset. The time also includes time spent on parsing the input. HermiT, Pellet and FaCT++ were called through OWL API-3.1, so the same parser was used for them. Jena reasoner was used in its native environment with the Jena parser.

In the early beginning of the experiment we had to exclude the FaCT++ reasoner from both tests. It turned out that FaCT++ does not work with rules [12] and it did not return any result instances. All the remaining reasoners strictly agreed on the results and returned the same sets of instances.

[1]This version of the corpus comes from the Dot.kom project's resources (http://nlp.shef.ac.uk/dot.kom/resources.html 2011-08-09 page, 2006-12-31 dataset).

Table II
TIME PERFORMANCE OF TESTED REASONERS ON BOTH DATASETS.

reasoner	**czech_fireman**	stdev	**acquisitions-v1.1**	stdev
Jena	161 s	0.226	1259 s	3.579
HermiT	219 s	1.636	$\gg$ 13 hours	
Pellet	11 s	0.062	503 s	4.145
FaCT++	Does not support rules.			

Time is measured in seconds. Average values from 6 measurements. Experiment environment: Intel Core I7-920 CPU 2.67GHz, 3GB of RAM, Java SE 1.6.0_03, Windows XP.

Also HermiT was not fully evaluated on the Acquisitions v1.1 dataset because it was too slow. The reasoner spent 13 hours of running to process only 30 of 600 files of the dataset. And it did not seem useful to let it continue.

D. Evaluation Results of the Experiment

Table II summarizes results of the experiment. The standard deviations are relatively small when compared to the differences between the average times. So there is no doubt about the order of the tested reasoners. Pellet performed the best and HermiT was the slowest amongst the tested and usable reasoners in this experiment.

From the results we can conclude that similar tasks can be satisfactorily solved by contemporary OWL reasoners because three of four tested reasoners were working correctly and two reasoners finished in bearable time.

On the other hand even the fastest system took 8.5 minutes to process 113 rules over 126MB of data. This is clearly significantly longer than a bespoke system would require. Contemporary Semantic Web reasoners are known still to be often quite inefficient and the experiment showed that using them today to do information extraction will result in quite poor performance. However, efficiency problems can be solved and in the context of Linked Data providing shareable descriptions of information extraction rules may be valuable.

VI. DISCUSSION

In this paper (Section IV-A), we have described a method how to apply an extraction ontology to a document ontology and obtain so called "annotated" document ontology. To have an "annotated" document ontology is almost the same as to have an annotated document. An annotated document is useful (easier navigation, faster reading and lookup of information, possibility of structured queries on collections of such documents, etc.) but if we are interested in the actual information present in the document, if we want to know the facts that are in a document asserted about the real word things then an annotated document is not sufficient. But the conversion of an annotated document to the real world facts is not simple. There are obvious issues concerning data integration and duplicity of information. For example when in a document two mentions of people are annotated as 'injured', what is then the number of injured people in the corresponding accident? Are the two annotations in fact linked to the same person or not?

In the beginning of our work on the idea of shareable extraction ontologies we planned to develop it further, we wanted to cover also the step from annotated document ontologies to the real world facts. The extraction process would then end up with so called "fact ontologies". But two main obstacles prevent us to do that.

1) Our IE engine is not yet capable to solve these data integration and duplicity of information issues and the real world facts would be quite imprecise then.
2) There are also technology problems of creating new facts (individuals) during reasoning.

A. SPARQL Queries – Increasing Performance?

There is also a possibility to transform the extraction rules to SPARQL construct queries. This would probably rapidly increase the time performance. However a document ontology would then have to exactly fit with the schema of the extraction rules. This would be a minor problem.

The reason why we did not study this approach from the beginning is that we were interested in extraction *ontologies* and SPARQL queries are not currently regarded as a part of an ontology and nothing is suggesting it to be other way round. Anyway the performance comparison remains a valuable task for the future work.

B. Contributions for Information Extraction

The paper combines the field of ontology-based information extraction and rule-based reasoning. The aim is to show a new possibility in usage of IE tools and reasoners. In this paper, we do not present a solution that would improve the performance of IE tools.

We also do not provide a proposal of a universal extraction format (although a specific form for the rule based extraction on dependency parsed text could be inferred). This task is left for the future if a need for such activity emerges.

VII. Conclusion

In the beginning of the paper we pointed out the draw back of so called extraction ontologies – in most cases they are dependent on a particular extraction/annotation tool and they cannot be used separately.

We extended the concept of extraction ontologies by adding the shareable aspect and we introduced a new principle of making extraction ontologies independent of the original tool: the possibility of application of an extraction ontology to a document by an ordinary reasoner.

In Section IV we presented a case study that shows that the idea of shareable extraction ontologies is realizable. We presented implementation of an IE tool that exports its extraction rules to an extraction ontology and we demonstrated how this extraction ontology can be applied to a document by a reasoner. Moreover, in Section V, an experiment with several OWL reasoners was presented. The experiment evaluated the performance of contemporary OWL reasoners on IE tasks (application of extraction ontologies).

A new publically available benchmark for OWL reasoning was created together with the experiment. Other reasoners can be tested this way.

Acknowledgments: This work was partially supported by Czech projects: GACR P202/10/0761, GACR-201/09/H057, GAUK 31009 and MSM-0021620838.

References

[1] D. W. Embley, C. Tao, and S. W. Liddle, "Automatically extracting ontologically specified data from html tables of unknown structure," in *ER*, ser. LNCS, vol. 2503. Springer, 2002, pp. 322–337.

[2] M. Labský et al., "The Ex Project: Web Information Extraction Using Extraction Ontologies," in *Knowledge Discovery Enhanced with Semantic and Social Information*, ser. Studies in Comput. Intellig. Springer, 2009, vol. 220, pp. 71–88.

[3] D. C. Wimalasuriya and D. Dou, "Ontology-based information extraction: An introduction and a survey of current approaches," *Journal of Information Science*, vol. 36, no. 3, pp. 306–323, June 2010.

[4] B. Yildiz and S. Miksch, "ontoX - a method for ontology-driven information extraction," in *ICCSA'07 - Part III*. Springer, 2007, pp. 660–673.

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

[6] D. W. Embley, "Toward semantic understanding: an approach based on information extraction ontologies," in *ADC '04*. Darlinghurst: ACS, 2004, pp. 3–12.

[7] B. Parsia, E. Sirin, B. C. Grau, E. Ruckhaus, and D. Hewlett. (2005) Cautiously approaching SWRL. Preprint submitted to Elsevier Science. 2011-08-09. [Online]. Available: http://www.mindswap.org/papers/CautiousSWRL.pdf

[8] J. Dědek, "Towards semantic annotation supported by dependency linguistics and ILP," in *ISWC2010*, ser. LNCS, vol. 6497. Springer, 2010, pp. 297–304.

[9] Z. Žabokrtský, J. Ptáček, and P. Pajas, "TectoMT: Highly modular MT system with tectogrammatics used as transfer layer," in *Proceedings of the 3rd Workshop on Statistical Machine Translation*. Columbus, OH, USA: ACL, 2008, pp. 167–170.

[10] B. Glimm, M. Horridge, B. Parsia, and P. F. Patel-Schneider, "A Syntax for Rules in OWL 2," in *OWLED 2009*, vol. 529. CEUR, 2009.

[11] B. Motik, U. Sattler, and R. Studer, "Query answering for owl-dl with rules," *Web Semantics*, vol. 3, pp. 41–60, July 2005.

[12] Wikipedia article: Semantic reasoner - reasoner comparison. 2011-08-09. [Online]. Available: http://en.wikipedia.org/wiki/Semantic_reasoner#Reasoner_comparison