Semantic Annotation as Design Recovery

Nadzeya Kiyavitskaya[1], Nicola Zeni[1], James R. Cordy[2], Luisa Mich[1] and John Mylopoulos[3]

[1] Department of Information and Communication Technology, University of Trento via Sommarive, 14, I-38050 Povo, TN, Italy,
{nadzeya.kiyavitskaya, nicola.zeni, luisa.mich}@dit.unitn.it
[2] School of Computing, Queens University, Kingston, Ontario, Canada K7L 3N6,
cordy@cs.queensu.ca
[3] Bahen Centre for Information Technology, University of Toronto, Ontario, Canada M5S 2E4,
jm@cs.toronto.edu

Abstract. Semantic annotations will play a crucial role in making the vision of the Semantic Web a reality. In this work, we propose a general domain-independent architecture for semantic markup adopting software design recovery techniques, and demonstrate its feasibility in a limited but realistic domain. The results of this experimentation are validated using three stage evaluation schema.

1 Introduction

Semantic Annotation (SA) is the process of inserting tags in a document to assign semantics to text fragments. SA is one way to create meaningful, maintainable, accessible documents for the Semantic Web. SA has many potential applications in such fields as Information Extraction, Question Answering, Text Summarization, Knowledge Management and Semantic Web.

SA is a challenging problem which is related to Natural Language Processing (NLP) field. However, traditional NLP techniques need to be revised, to take advantage of software engineering methods and technologies [Leidner2003] [Boguraev1995]. Problematic issues in NLP applications for semantic annotation from a software engineering perspective include:

- *Accuracy.* All NLP tools have so called incompleteness property, i.e. they can never guarantee to provide all and only correct results.
- *Flexibility.* NLP systems need to be able to handle different data formats (newspaper, e-mails, HTML, etc.).
- *Robustness.* This feature characterizes the viability of a system under abnormal conditions. In particular, in NLP, it means stability to different text types or domains.
- *Scalability.* Space and run time limitations must be overcome.
- *Data Sparseness.* Many NLP systems rely on the availability of training resources, which are typically expensive to develop.

– *Complexity.* Average complexity must be estimated. Long response time can render a system unacceptable for human users.
– *Multilinguality.* Independence from character encodings, lexicographic sorting orders, display of numbers, dates etc. needs to be ensured.
– *Testing and evaluation.* System performance must be estimated.

Depending on the resources and knowledge base embedded in an NLP tool, we can classify such tools into two categories:

– Heavyweight tools: are capable of full text understanding with different NLP modules, but require huge investments in resources and time, and are hardly scalable to large quantities of documents;
– Lightweight tools: achieve good performance for a particular application with small investment.

Both approaches have their strengths and weaknesses. Lightweight methods are usually ad hoc and application-specific. On the opposite side, heavyweight systems are expensive and do not scale. Given the number and scope of documents on the world-wide web, transition to the semantic web vision cannot be achieved without large-scale efficient automation on semantic markup [Dill2003].

The objective of our project is to develop a methodology supported by a set of lightweight tools for semi-automatic Semantic Annotation of text. In this paper, we present our approach based on a design recovery technique which has been already proven efficient and scalable in software analysis area. We describe a preliminary experiment in applying the same technical solutions and the framework to evaluate the quality of annotations.

The structure of the rest of the paper is as follows: section 2 introduces text processing with TXL, section 3 provides the details of our approach to SA process, section 4 describes the setup of experimentation, section 5 presents evaluation framework and results, section 6 reviews related projects in the SA field, and the final section summarizes the work done so far and outlines directions for future research.

2 Text Processing with TXL

In our approach the technique used for text analysis is the compiler parsing technology TXL[1]. TXL is a programming language for expressing solutions using structural source transformation from input to output. The structure imposed on input is specified by an ambiguous context free grammar. Transformation rules are then applied, and transformed results returned to text. TXL uses full backtracking with ordered alternatives and heuristic resolution which allows efficient, flexible, general parsing. Grammars and transformation rules are specified by example.

The transformation phase can be considered as term rewriting, but under

[1] http://www.txl.ca

functional control. Functional programming control provides abstraction, parametrization and scoping. TXL allows grammar overrides to extend, replace and modify existing specifications. Grammar overrides can be used to express *robust parsing*, technique to allow errors or exceptions in input not explained by grammar. Overrides can also express *island grammars*. Island parsing recognizes interesting structures, "islands", embedded in a "sea" of uninteresting or unstructured background. TXL also supports *agile parsing* - customization of the parse to each individual transformation task. This is a simple example of TXL program realizing island parsing paradigm:

```
% Input is a sequence of items
    redefine program
        [repeat item]
    end redefine

% Items are either interesting or uninteresting
define item
    [declaration_or_statement]
    | [uninteresting]
end define

define uninteresting
    [token] | [key]     % TXL idiom for "any input item"
end define

% Transform aspect only; rest of input remains the same
rule main
    replace $ [declaration_or_statement]
        Code[declaration_or_statement]
    by
        Code [prettyFormat]
end rule
```

Originally, TXL was designed for experimenting with programming language dialects, but soon it was realized useful for many other tasks, such as static analysis (an important application is design recovery), interpreters, preprocessors, theorem provers, source markup, XML, language translation, web migration, etc. Moreover, TXL was successfully used for applications other than programs: handwritten math recognition, document table structure recognition [Zanibbi2002] [Zanibbi2004]; business card understanding [Oertel2004].

TXL has been already proven as an efficient instrument to help software engineers in reverse engineering and in particular, in the design recovery task.

Software reverse engineering or *program comprehension* is the process of identifying software components, their inter-relationships and representing these entities at a higher level of abstraction [Nelson1996]. Software reverse engineering can be also combined with conceptual modeling of the source code. *Design*

recovery is a specialization of the reverse engineering and it entails the static analysis of the source code of a (large) software system to identify entities and relationships according to a software design model (entity-relationship (ER) schema). The result is normally a design database, plus an annotated version of the source code marked up with design relationships. Software design recovery has been highly successful at both technical and business-level semantic markup of large scale software systems written in a wide variety of programming languages.

We propose to use TXL as the basis of a new lightweight method for SA. This choice is motivated by the fact that document analysis for the Semantic Web has much in common with design recovery and it poses similar problems:

- the need for robust parsing techniques because real documents do not always match the grammars of the languages they are written in;
- the need to understand the semantics of the source text according to a semantic model;
- semantic clues drawn from a vocabulary of the semantic domain;
- contextual clues drawn from the syntactic structure of the source text;
- inferred semantics from exploring relationships between identified semantic entities and their properties, contexts and related other entities.

TXL addresses such problems in a fast and flexible way and allows scalable processing of large documents.

3 Semantic Annotation Methodology

Our methodology is grounded on the design recovery process which typically contains three main passes:

1. Lightweight robust parse to get basic structure, transformation rules use vocabulary and structural patterns to infer source markup of basic facts;
2. Facts are externalized to database for inference;
3. Transformation rules use inferences and structural patterns to infer semantic markup of design facts, marked-up source is ready for design-aware transformations.

We designed a tool that performs SA in similar manner (see fig. 1).

The input of the system consists of textual documents and a conceptual schema. The conceptual schema can be a part of existing domain ontology. Entities of the schema are used to generate tags for annotation.

The workflow has two main phases:

1. First phase consists of lightweight parsing and semantic markup of basic facts (semantic markup is based on semantic model of the domain in combination with syntactic patterns);
2. Second phase is externalization of the facts to database, which can be then used by search engine for queries.

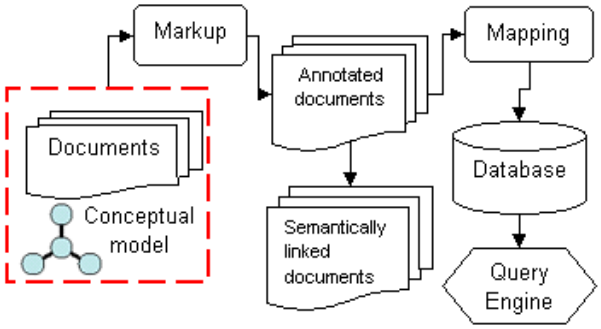

Fig. 1. Workflow

The transformations of the last third stage are not yet implemented in our tool, but we plan to apply them further to filter marked up facts according to the constraints imposed by the conceptual schema.

In the next section all the stages of processing are explained in detail on the example of tourism domain application.

4 Experimentation

To demonstrate the work of our method we present the results of a preliminary experimentation. Thus far experiments have been completed for only one domain because the development of generic methodology, tuning of the tool and design of the evaluation framework required more time than we expected. However, this experiment is only the beginning and we plan to recover from this drawback in future work.

We have been working in the domain of travel documents: descriptions of accommodations in the popular tourist destinations. The goal is to provide relevant information for the users reading these texts, such as, for example, location and price of the accommodation, term of availability, and so on. For this purpose, the accommodation advertisements must be marked up according to accommodation ontology containing these items (see fig.3), which was further reduced by hand to an entity-relationship schema in XML format for input to our system.

As input documents for SA task we considered accommodation advertisements drawn from an existing Italian web site. Here is one example of an advertisement (fig.4).

In order to make a realistic test of the generality of the method, we restricted ourselves to some constraints: no proper nouns or location-dependent phrases in our vocabulary, raw uncorrected text, and no formatting or structural cues. Following the methodology described in the previous section, for this experiment we adopted the same multi-level process.

The architecture explicitly factors out reusable domain-independent knowledge such as the structure of basic entities and language structures, shown on

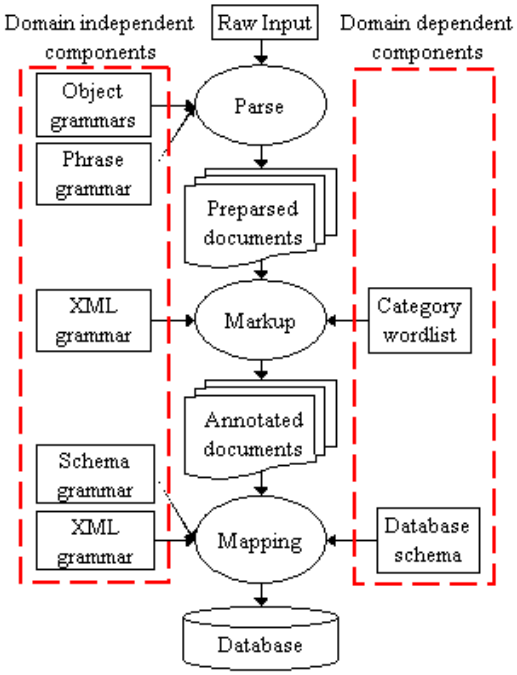

Fig. 2. Methodology

the left hand side, while allowing for easy change of semantic domain, characterized by vocabulary and conceptual schema, shown on the right. This way we facilitate reusability of the tool over different domains and document formats.

First, text is parsed into sentences (can be any other text unit) and basic objects are detected. These objects usually can be described by a small set of patterns and then reused over different domains. So far our list of objects includes e-mail addresses, web addresses, phone numbers, dates and prices. For instance, the grammar for phone numbers is represented in the following way:

```
% Part of price grammar
tokens
    number  "\d\d*"
end tokens

define money
 [amount] [opt hyphen_amount] [space_currency]
 | [currency] [opt ':] [opt space] [amount] [repeat hyphen_amount]
 | [repeat number_dot] [anynumber] [dot_zerozero] [opt space_currency]
end define
```

The phrase grammar block carries structural information how to delimit text units that we want to markup. This unit can be a short phrase, a sentence

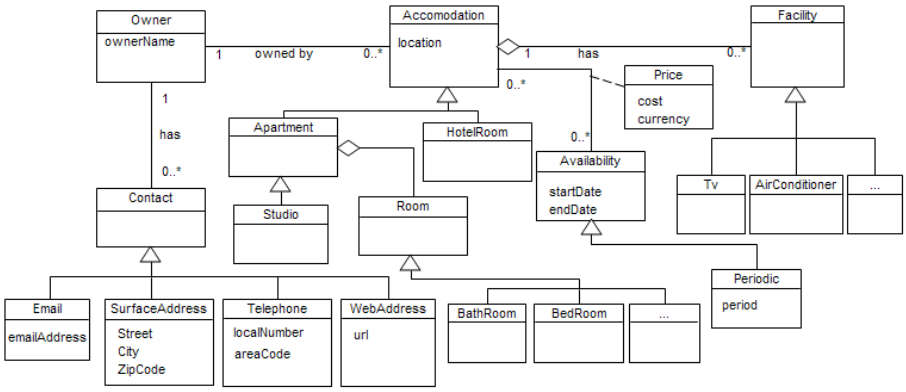

Fig. 3. Conceptual schema

Fig. 4. Sample of advertisement

or whole paragraph depending on the required granularity of annotation. In our experiments with accommodation advertisements we used sentence grammar because even if the text is short the user is interested in complete answer.

Then using the objects found on the previous stage and checking for the presence of category keywords, the related phrases are identified and marked up. XML grammar component used at the Markup phase is actually the grammar of tags for inserting markup into documents (i.e. grammar for XML open tag: '<[identifier]'>, for XML closing tag '</[identifier]'>). Category wordlist is domain dependent component including set of categories and keyword lists corresponding to each category. Keyword list consists of positive markers (simply one word or combination of words, as for example "Information System") and negative markers. If for a given category any of the positive markers are detected within a text unit (sentence in current experiment), then the unit is tagged under this category, unless any of negative markers is found.

Annotated documents are provided to the Mapping phase which is responsible for filling the database with annotations. This phase uses domain dependent component - the database schema. Domain independent component of this stage are schema grammar and XML grammar. Schema grammar is used for reading the database schema from file, and XML grammar for extracting markups from

the output documents of the previous stage. Finally XML markups are mapped into correspondent fields of the database. It is important to emphasize that "complex" text processing (i.e. objects recognition, sentence delimiting) is made only once at the first phase, and never repeated again. All the following phases perform fast superficial processing using very simple grammars.

5 Evaluation Framework

The choice of evaluation method to verify the quality of automatic annotation is an additional difficulty. For this purpose we specially designed a three steps evaluation framework.

In the first step we compared the system output directly with human annotations (2 persons were involved). We assume that human performance is the upper-bound for automatic language analysis. However, this type of evaluation cannot be applied on the large scale, because obviously we cannot afford human annotators tagging gigabytes of text. Also in this case, we must take into account annotators' disagreement and in order to obtain realistic evaluation we must "calibrate" system performance relative to human performance. For this purpose we calculated not only the system performance against each manual annotation, but also the performance of each human annotator against the other. Subtracting the difference between the performances we can conclude how much of human work could be actually done by the tool.

In the second step, we check if the use of automatic tool increases the productivity of human annotators. We noted the time used for manual annotation of the original textual documents and compared it to the time used for manual correction of the automatically annotated documents. The percentage difference of these two measures shows how much time can be saved when the human annotator is assisted by the tool.

Our third step compares system results against the human corrections done in the previous step. The distinction of this phase from the first one is that when a human annotator works directly on the original document he/she can make errors or miss some items because of the lack of attention; while working on the document already annotated by the tool he/she can easily note the defects and therefore produces a higher quality annotation.

5.1 Evaluation against Human Annotation

This evaluation involves comparison of system results against manually annotated texts. It involved two annotators and so far is done for sentences (coarse-grained annotation). We applied the following metrics: recall - R, precision - P, fallout - F, accuracy - Acc, error - Err [Yang1999]:

$$R = \frac{TP}{TP + FN} \tag{1}$$

$$P = \frac{TP}{TP + FP} \tag{2}$$

$$F = \frac{FP}{FP + TN} \tag{3}$$

$$Acc = \frac{TP + TN}{N} \tag{4}$$

$$Err = \frac{FP + FN}{N} \tag{5}$$

The total number of test items is N = TP + FP +FN +TN, where:

- TP is the number of items correctly assigned to this category (True Positives);
- FP is the number of items incorrectly assigned to this category (False Positives);
- FN is the number of items incorrectly rejected from this category (False Negatives);
- TN is the number of items correctly rejected from this category (True Negatives).

We also calculated F-measure which is harmonic mean of recall and precision:

$$F{-}measure = \frac{2 \times R \times P}{R + P} \tag{6}$$

Table 1 represents the results of system performance compared directly with human annotation.

Table 1. System performance against human annotation

Measure	Sys vs Human2	Sys vs Human1	Average
Recall	92.42%	92.19%	92.31%
Precision	76.25%	73.75%	75.00%
Fallout	7.54%	8.27%	7.90%
Accuracy	92.45%	91.82%	92.14%
Error	7.55%	8.18%	7.86%
F-Measure	83.56%	81.94%	82.76%

Tables 2 and 3 represent the results of system performance compared to relative human performance.

We can see that compared to humans, the system annotates more items, but with less precision. This test shows that the tool has potential to assist a human annotator in generating preliminary markup, leaving further correction to the human.

In order to estimate the overall difference between human and system performances we should observe the values of an aggregate characteristic F-measure. The obtained differences are 5.67–7.29%, which shows that system was able to complete approximately 92.71–94.33% of human work.

Table 2. Calibrating system performance against Human1

Measure	Human2 vs Human1	Sys vs Human1	Difference
Recall	90.63%	92.19%	-1.56%
Precision	87.88%	73.75%	14.13%
Fallout	3.15%	8.27%	-5.12%
Accuracy	95.60%	91.82%	3.77%
Error	4.40%	8.18%	-3.77%
F-Measure	89.23%	81.94%	7.29%

5.2 Productivity Measures

During this step of evaluation we compared the times spent by annotator to perform annotation "from the blank page" and to correct system annotation. The results showed that the tool saved about 78% of the annotator's time on our sample set of data. Thus with an appropriate interface for doing corrections easily, the time savings would likely be significantly greater than we observed.

5.3 Evaluation against Human Annotation Correcting System

On this step system performance is calculated considering manual corrections of automatic annotation as the Ground Truth (see Table 4).

Comparing the results on different test sets the performance decreases slightly, only for the last test set Recall value changes significantly. This shortcoming can be explained by the high dissimilarity of these documents to the training documents by content and structure. For example, in this case location names were represented as a string with exact postal address before textual description of the accommodation. This maid it difficult to detect location phrases, especially because in our experiment for the sake of generality we did not use any gazetteer.

At best what we can say is that the results of our small study is that there is a real potential for a method based on the software design techniques. Even without local knowledge and using a very small vocabulary, we have been able to demonstrate accuracy comparable to the best heavyweight methods, albeit thus far for a very limited domain. Performance of our as yet untuned experimental

Table 3. Calibrating system performance against Human2

Measure	Human1 vs Human2	Sys vs Human2	Difference
Recall	87.88%	92.42%	-4.55%
Precision	90.63%	76.25%	14.38%
Fallout	2.38%	7.54%	-5.16%
Accuracy	95.60%	92.45%	3.14%
Error	4.40%	7.55%	-3.14%
F-Measure	89.23%	83.56%	5.67%

Table 4. System performance against manually corrected annotations

Measure	Rome (10 ads) Training set	Rome (10 ads) Test set-1	Rome (100 ads) Test set-1	Venice(10 ads) Test set-2
Recall	98.73%	94.20%	92.31%	86.08%
Precision	97.50%	97.01%	96.93%	95.77%
Fallout	0.84%	0.87%	0.93%	1.29%
Accuracy	99.06%	98.00%	97.43%	95.51%
Error	0.94%	2.00%	2.57%	4.49%
F-Measure	98.11%	95.59%	94.56%	90.67%

tool is also already very fast, handling 100 advertisements for example in about 1 second on a 1 GHz PC.

6 Related Work

The development of different Semantic Web applications became recently the area of the intensive research work. In this review we list some of these tools and consider the methodologies used.

One of the first attempts to realize SA was done with the SHOE system [Sean97] which allowed Web page authors to manually annotate their Web documents with machine-readable knowledge. Another pioneering tool assisting to annotate documents manually was Ontobroker [Decker99]. AeroDAML tool [Kogut2001] applied natural language information extraction techniques to automatically generate DAML annotations from Web pages. Pankow a project of Karlsruhe University [Cimiano04] proposes a methodology to fully automate document annotation by using statistical and pattern-matching techniques to automatically discover relevant concepts in the document. Among the variety of projects we are going to note some of the most relevant to our approach.

SemTag [Dill2003] is an application that performs automated semantic tagging of large corpora. It is based on the Seeker platform for large-scale text analysis. It tags large numbers of pages with terms from a standard ontology. As a centralized application it can use corpus statistics to improve the quality of tags. So far, the TAP knowledge base has been used as a standard ontology. TAP contains lexical and taxonomic information about: music, movies, authors, sports, autos, health, and other popular objects. RDFS is used as a language for representing the annotations. Current focus is detecting the occurrence of particular entities in web pages. This task requires also disambiguation step.

– Methodology. SemTag flow consists of the following steps: 1) Spotting pass: documents are retrieved, tokenized, and then processed to find instances of approximately 72KB labels of TAP taxonomy. 2) Learning pass: sample of data is scanned to determine distribution of terms. 3) Tagging pass: each reference is disambiguated and a record is inserted into a database.

– Evaluation. Large scale evaluation: 264 million web pages, 550 million generated and automatically disambiguated semantic tags. Of these labels approximately 79% are judged to be on-topic. 750 human judgments are used as a training set for the algorithm; other 378 human judgments are applied to estimate the performance. The hardware requirements of the system are extremely high. It consists of 128 dual processor 1GHz machines. I/O speed is 3MB/sec on a single 1GHz processor. The total time taken to process the web is 32 hours.

The KIM (Knowledge and Information Management) platform [Kiryakov2005] was designed to demonstrate the implementation of the SA vision. KIM is an application for automatic ontology-based named entities annotation, indexing and retrieval. It is based on GATE (General Architecture for Text Engineering), University of Sheffield [2]. It uses lightweight upper-level ontology (KIMO) focused on the named entity classes (consisting of about 250 classes and 100 properties) encoded in RDF(S). Also KIM has a knowledge base of approximately 80,000 entities of general importance to allow information extraction (IE) on inter-domain web-content.

– Methodology. IE is grounded on the GATE framework. The processing goes through several steps, such as tokenization, splitting to sentences, part of speech tagging. A Semantic Gazetteer uses the KB to generate lookup annotations. Ontology aware pattern-matching grammars allow precise class information to be handled via rules at the optimal level of generality. The grammars are used to recognize NE, with class and instance information referring to the KIM ontology and KB. Recognition of identity relations between the entities is used to unify their references to the KB. Based on the recognized NE, template relation construction is performed via grammar rules. As a result of the latter, the KB is being enriched with the recognized relations between entities. At the final phase of the IE process, previously unknown aliases and entities are being added to the KB with their specific types.
– Evaluation. The evaluation [Popov2003] of the approach was performed only for flat NE types (date, person, organization, location, percent, money). The authors explain this drawback by the lack of test data and evaluation metrics. However, the performances reported for this experiment are very encouraging (for all entities average Recall is 84%, Precision is 86%). System requirements [Popov2004]: enough to have Pentium 4 (2.53 GHz) PC to acquire the following performance metrics: annotation - 8 kb/s; indexing - 27 kb/s; storage - 6 kb/s. The times growth has a logarithmic dependency for bigger documents.

In KIM, as well as in SemTag, SA is considered as the process of assigning to the entities in the test links to their semantic descriptions, provided by ontology. This means that the focus of the analysis is mainly named entities recognition

[2] http://gate.ac.uk

or detecting instances of the universals (concepts, classes, relations, attributes).

S-CREAM (Semi-automatic CREAtion of Metadata) provides an annotation and authoring framework that integrates a learnable information extraction component [Handschuh2002].

- Methodology. A domain ontology can be the basis for the annotation of different types of documents. The user have do define which part of the ontology is relevant for the learning task. The user can perform a crawl to collect the necessary documents. Then users have too manually annotate a corpus for training the learner. Text is preprocessed using Annie, shallow IE system included in Gate package (text tokenization, sentence splitting, part of speech tagging, gazetteer lookup and named entity recognition). Each document of the corpus is processed by learning plugin which generates extraction rules. Then the induced rules are applied for semi-automatic annotation.
- Evaluation. No evaluation is provided in publications.

Our work differs from all of these approaches in three fundamental ways - first, it uses an extremely lightweight but robust context-free parse in place of tokenization and part-of-speech recognition. Also our tool don't have the learning phase, instead it has to be tuned manually when being ported to a particular application, substituting or extending domain dependent components. Second, it does not require a gazetteer or knowledge base of known proper entities, rather it infers their existence from their structural and vocabulary context, in the style of software analyzers. And third, it has already been shown to handle not only flat entities higher-level semantic markup for concepts above and depending on entities rather than just the entities themselves.

7 Conclusions and Future Work

In this work we have demonstrated that applying software design recovery techniques to semantic annotation of documents is feasible and has potential. It is also clear that these techniques can retain their efficiency in new domain, exhibiting very fast and linear performance even without tuning. We consider as contribution of our work also the evaluation schema that we designed to measure the quality of annotation.

The future work includes testing and validation of our method on large corpora and richer conceptual spaces so that a more meaningful comparison with the state of the art can be done. There are still a number of techniques used in software analysis that we have not taken advantage of: alias resolution, unique naming, architecture patterns, markup refinement and so on. In future we plan to explore these other techniques to improve the quality our semantic annotation.

References

[Boguraev1995] Boguraev, B.K., Garigliano, R., Tait, J.I.: Editorial, Natural Language Engineering **1(1)** 1–7, 1995, ISSN: 1351-3249(199503)1:1;1-F

[Cimiano04] Cimiano, P., Handschuh, S., and Staab, S.: Towards the self-annotating web. In *Proceedings of the 13th international conference on World Wide Web*, 462–471, ACM Press, 2004

[Decker99] Decker, S., Erdmann,M., Fensel,D., and Studer, R.: Ontobroker: Ontology based access to distributed and semi-structured unformation. In *DS-8: Database Semantics - Semantic Issues in Multimedia Systems, IFIP TC2/WG2.6 Eighth Working Conference on Database Semantics*, 351–369, Rotorua, New Zealand, 1999

[Dill2003] Dill, S., Eiron, N., Gibson, D., Gruhl, D., Guha, R., Jhingran, A., Kanungo, T., McCurley, K. S., Rajagopalan, S., Tomkins, A., Tomlin, J.A., Zien, J. Y.: A Case for Automated Large-Scale Semantic Annotation. *Journal of Web Semantics*, **1(1)** 115–132, 2003

[Handschuh2002] Handschuh, S., Staab, S., Ciravegna, F.: S-CREAM Semi-automatic CREAtion of Metadata. The 13th International Conference on Knowledge Engineering and Management (EKAW2002), ed. Gomez-Perez, A., Springer Verlag, 2002

[Cordy2004] Cordy, J.: A Language for Programming Language Tools and Applications. *Proc. of LDTA 2004 ACM 4th Int. Workshop on Language Description, Tools and Applications*, Barcelona, April, 2004

[Kiryakov2005] Kiryakov, A., Popov, B., Terziev, I., Manov, D., Ognyanoff, D.: Semantic Annotation, Indexing, and Retrieval. *Elsevier's Journal of Web Sematics*, **2(1)**, 2005

[Kiyavitskaya 2004] Kiyavitskaya, N., Zeni, N., Mich, L. and Mylopoulos, J.: Experimenting with Linguistic Tools for Conceptual Modelling: Quality of the models and critical features. *Proc. of the NLDB2004*, **3136** 135–146, 2004

[Leidner2003] Leidner, J. L.: Current Issues in Software Engineering for Natural Language Processing. *Proc. of the Workshop on Software Engineering and Architecture of Language Technology Systems (SEALTS)*, the Joint Conf. for Human Language Technology and the Annual Meeting of the Noth American Chapter of the Association for Computational Linguistics (HLT/NAACL'03), Edmonton, Alberta, Canada, 45–50

[Kogut2001] Kogut, P. and Holmes, W.: AeroDAML: Applying Information Extraction to Generate DAML Annotations from Web Pages. *First International Conference on Knowledge Capture (K-CAP 2001)*. Workshop on Knowledge Markup and Semantic Annotation, Victoria, B.C., Canada, October 2001.

[Nelson1996] Nelson, M.L.: A Survey of Reverse Engineering and Program Comprehension, 1996

[Oertel2004] Oertel, C., O'Shea, S., Bodnar, A. and Blostein, D.: Using the Web to Validate Document Recognition Results: Experiments with Business Cards. *Proc. of the SPIE*, **5676** 17–27, 2004

[Popov2003] Popov, B., Kiryakov, A., Ognyanoff, D., Manov, D., Kirilov, A., Goranov, M.: Towards Semantic Web Information Extraction. *Human Language Technologies Workshop at the 2nd International Semantic Web Conference (ISWC2003)*, 20 October 2003, Florida, USA

[Popov2004] Popov, B., Kiryakov, A., Ognyanoff, D., Manov, D., Kirilov, A.: KIM - a semantic platform for information extaction and retrieval. *Journal of Web Semantics*, **10(3-4)**, 2004, 375–392, Cambridge University Press

[Sean97] Sean, L., Lee, S., Rager, D., and Handler, J.: Ontology-based web agents. *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, eds. Johnson, W.L., and Hayes-Roth, B., 59–68, Marina del Rey, CA, USA, 1997, ACM Press

[Yang1999] Yang, Y.: An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval*, 1999, **1(1-2)**, 67–88

[Zanibbi2002] Zanibbi, R., Blostein, D., Cordy, J.R.: Recognizing Mathematical Expressions Using Tree Transformation, IEEE Transactions on Pattern Analysis and Machine Intelligence, **24 (11)**, 1455–1467, 2002

[Zanibbi2004] Zanibbi, R., Blostein, D., Cordy, J.R.: A Survey of Table Recognition: Models, Observations, Transformations, and Inferences. *International Journal of Document Analysis and Recognition*, **7(1)**, 1–16, Sept. 2004