Semi-Automatic Semantic Annotations for Web Documents

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 annotation of the web documents is the only way to make the Semantic Web vision a reality. Considering the scale and dynamics of worldwide web, the largest knowledge base ever built, it becomes clear that we cannot afford to annotate web documents manually. In this work we propose a generic domain-independent architecture for semi-automatic semantic annotation, basing on the lightweight and robust techniques, proven effective in source code processing for software analysis field. We demonstrate feasibility of our method applying it for annotation of the documents for Tourism domain. The results of this experiment are validated using a three-stage evaluation scheme.

1 Introduction

Semantic annotation is the process of inserting tags in a document to assign semantics to text fragments allowing to create the documents processable not only by humans but also automated agents. However, considering the scale and dynamics of worldwide web, application of the traditional natural language processing techniques to annotate documents semantically must be revised. From the engineering perspective there is a number of requirements important to be faced when designing a text processing system [Leidner2003] [Boguraev1995]:

- *accuracy*: performance must be estimated to access the ability of the tool to retrieve all and only correct answers;
- *flexibility and robustness*: these features characterize the viability of a system under abnormal conditions and stability to different text types or domains;
- *scalability*: space and run time limitations must be overcome;
- *data sparseness:* dependence on expensive training resources can be an obstacle for porting the tool in a different domain;

– *complexity*: 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.

Similar problems are faced up in the software analysis field when developing tools for design recovery, source code analysis and markup. For this reason, we can exploit a large heritage gained by this field, which developed a number of the techniques enable to process billions lines of the source code. We propose to apply these methods as a base of new lightweight tool for semi-automatic semantic annotation of web documents. In the present paper we demonstrate a preliminary experiment on employment the same technical solutions in the domain of Tourism, we also describe the framework to evaluate the quality of annotations.

The paper is structured as follows: section 2 introduces text processing with TXL, section 3 provides the details of our approach, section 4 describes the setup of our first experiment, section 5 presents an evaluation framework and results of the experiment, section 6 reviews related projects in the field, and the final section summarizes the results and outlines directions for future work.

2 Semantic Annotation as Design Recovery

Our lightweight method for text processing is based on the technology proven efficient instrument to help software analysis area, and in particular, in reverse engineering and design recovery.

Software reverse engineering is the process of identifying engineers software components, their inter-relationships and representing these entities at a higher level of abstraction [Nelson1996]. This method can be also combined with conceptual modeling of the source code. Specialization of reverse engineering, *design recovery* implies the static analysis of the source code of a (large) software system to identify entities and relationships according to a software design model. The result is normally a design database and an 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.

Interestingly, two different domains, document analysis for the Semantic Web and design recovery of source code, pose 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.

We propose to use source analysis and transformation system TXL[1] as the basis of a new lightweight method for SA. TXL allows 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 process in TXL can be considered as term rewriting, but under 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

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

analysis, interpreters, preprocessors, theorem provers, source markup, 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].

3 Semantic Annotation Methodology

Our methodology is based on the LS/2000 software analysis architecture [Dean2001] which contains three 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 semantic annotation in similar manner (Fig 1).

The input of the system consists of textual documents and a conceptual

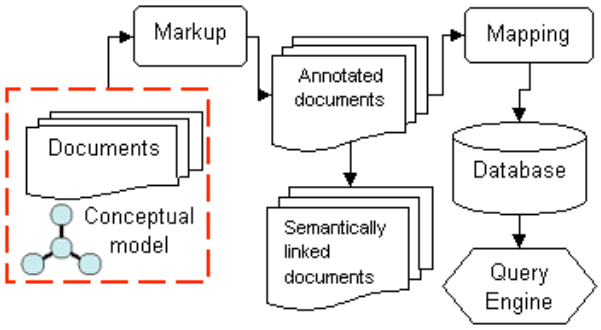

Fig. 1. The workflow of semantic annotation method based on the LS/2000 software analysis system

scheme. The conceptual scheme can be a part of existing domain ontology. Entities of the scheme 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 entities (email and web addresses, monetary formats, date and time formats, and so on) and language structures (object, document, paragraph, sentence and phrase structure);
2. Second phase is externalization of the facts to database, which can be then used by search engine for queries.

The transformations of the last third stage are not yet implemented in our tool, but we plan to explore this technique in future to verify if the quality of automatic annotations can be improved imposing constraints of the conceptual scheme.

Following section explains in detail all the stages of the algorithm on the example of application in Tourism domain.

4 Experiment Setup

To prove the viability of our method we present the results of our preliminary experiment in the domain of travel documents, in particular, accommodations ads of the popular tourist destinations in Italy (Fig 2).

Fig. 2. Example of an announcement retreived from the web site

The goal of this work was to provide the users browsing on-line ads with relevant information, such as, for example, location and price of the accommodation, availability, facilities provided, etc. For this purpose the accommodation ads must be marked up according to the domain conceptual model (Fig 3).

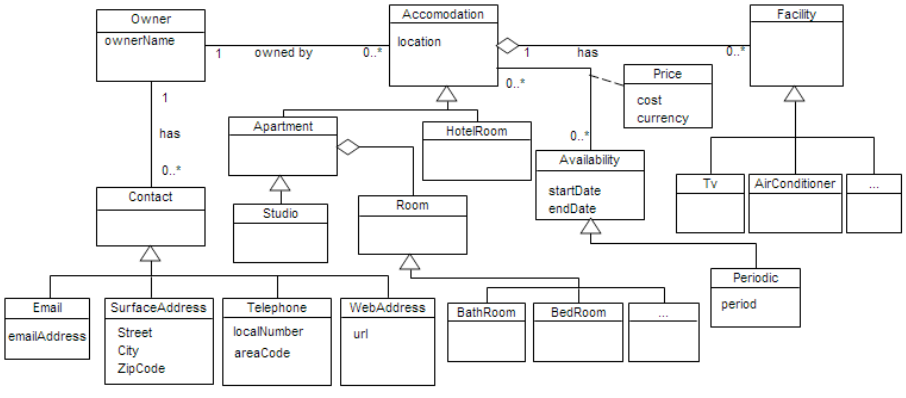

Fig. 3. Conceptual model of the domain of accommodation ads

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 (Fig 4).

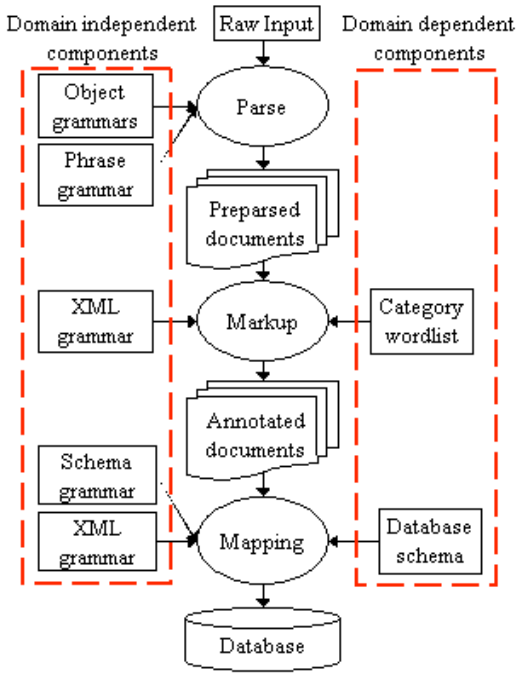

Fig. 4. Architecture of the semantic annotation process

The architecture explicitly factors out reusable domain-independent knowledge such as the structure of basic entities and language structures, shown on the left hand side, while allowing for easy change of semantic domain, characterized by vocabulary and conceptual scheme, shown on the right. This way we facilitate reusability of the tool over different domains and document types.

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 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 annotated under this category, unless any of negative markers is found.

Annotated documents are provided to the Mapping phase which fills the database scheme with annotations. Domain independent component of this stage are scheme grammar and XML grammar. Scheme grammar is used for reading the database scheme 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. 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.

To estimate the system performance we applied the following metrics (according to the definitions provided in [Yang1999]):

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

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

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

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

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

We also calculated *F-measure*, the harmonic mean of recall and precision:

$$F{-}measure = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{6}$$

Here N is the total number of the test items:

$$N = TP + FP + FN + TN \tag{7}$$

TP is the number of items correctly assigned to a category (true positives);
FP is the number of items incorrectly assigned to a category (false positives);
FN is the number of items incorrectly rejected from a category (false negatives);
TN is the number of items correctly rejected from a category (true negatives).

5.1 Evaluation against Human Annotation

In the first stage of evaluation the tool and each of two human annotators marked up a sample set of ten advertisements different from the training set used to tune the tool for the domain. The tool was then compared against each of the human markers for this set separately (Table 1), and then calibrated against each of the two as definitive (Table 2).

Table 1. System performance for each semantic category

Entity	Recall (A1)	Recall (A2)	Recall (avg)	Precision (A1)	Precision (A2)	Precision (avg)
Location	90,91%	90,91%	90,91%	100,00%	100,00%	100,00%
Facility	100,00%	87,50%	93,75%	47,83%	60,87%	54,35%
Price	100,00%	100,00%	100,00%	88,24%	82,35%	85,29%
Type	100,00%	100,00%	100,00%	60,00%	66,67%	63,33%
Term	50,00%	66,67%	58,33%	57,14%	57,14%	57,14%
Contact	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%

Table 2. Calibrating system performance against human annotators

Measure	A2 vs A1	Tool vs A1	A1 vs A2	Tool vs A2
Recall	90.63%	92.19%	87.88%	92.42%
Precision	87.88%	73.75%	90.63%	76.25%
Fallout	3.15%	8.27%	2.38%	7.54%
Accuracy	95.60%	91.82%	95.60%	92.45%
Error	4.40%	8.18%	4.40%	7.55%
F-Measure	89.23%	81.94%	89.23%	83.56%

In order to thoroughly compare human and system performances it is necessary to observe differences of F-measure, as it is an aggregate characteristic. The obtained variance is 5.67–7.29%, which leads us to conclusion that the tool was able to complete about 92.71–94.33% of human work.

5.2 Productivity Measures

During this stage 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

In the third stage, we gave the human annotators the advantage of correcting automatically marked up text to create their markups, and compared the final human markup to the original opinion of the tool (Table 3).

Table 3. 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%

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, which maid it difficult to detect phrases related to locations, especially because in our experiment for the sake of generality we did not use gazetteers.

Concluding this experimental study we can say is that the method based on the software design techniques has a potential. 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 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 exploited and system requirements declared.

One of the first attempts to allow semantic annotation of web documents was done with the SHOE system [Sean97], enabling web page authors to manually annotate their documents with machine-readable metadata. Another pioneering tool assisting to insert semantic markups in a manual way was Ontobroker [Decker99]. AeroDAML tool [Kogut2001] applied natural language information extraction techniques to automatically generate DAML annotations from web pages. Project of Karlsruhe University, Pankow [Cimiano04] uses statistical and

pattern-matching techniques to automatically discover relevant concepts in the document. Among the variety of tools we are going to underline several most recent projects oriented to a large-scale text markup.

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. The goal of SemTag annotator is to detect the occurrence of these entities in web pages.

- Methodology. SemTag flow consists of the following steps: 1) Spotting pass: documents are retrieved, tokenized, and then processed to find instances of approximately 72 ,KB labels of TAP knowledge base. 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. SemTag was evaluated on a set of 264 million web pages, the tool was able to generate and disambiguate 550 million semantic tags, approximately 79% of them were judged to be on-topic. During this experiment 750 human judgments were used as a training set for the algorithm and other 378 human judgments were applied to estimate the performance. System was realized on 128 dual processor 1GHz machines. The total time taken to process the web is 32 hours.

The KIM (Knowledge and Information Management) platform [Kiryakov2005] is an tool for automatic ontology-based named entities annotation, indexing and retrieval based on GATE (General Architecture for Text Engineering), University of Sheffield[2]. It uses lightweight upper-level ontology (KIMO) consisting of the named entity classes (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 on inter-domain web content.

- Methodology. KIM is build on the top of GATE architecture. Text processing in GATE is fulfilled in several steps, including a number of NLP techniques, such as tokenization, splitting to sentences and part-of-speech tagging. A semantic gazetteer is used 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 named entities with class and instance information referring to the KIM ontology and the knowledge base. Based on the recognized semantic constructions, template relation construction is performed by means of the grammar rules. As a result, the knowledge base is enriched with the recognized relations between entities. On the final phase, previously unknown aliases and entities are added to the knowledge base with their specific types.

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

– Evaluation. KIM was tested for flat named entities types [Popov2003]: date, person, organization, location, percent, monetary amounts, reporting high accuracy metrics for this experiment: average Recall – 84%, Precision – 86%. KIM platform requires Pentium 4 (2.53 GHz) computer to acquire the following performance rates: annotation – 8 kb/s; indexing – 27 kb/s; storage – 6 kb/s [Popov2004]. The time grows logarithmically depending on the size of input documents.

In KIM, as well as in SemTag, annotation is considered as the process of assigning to the entities in the test links to their semantic descriptions provided by ontology, therefore the focus of is mainly maid on recognition of named entities, categorization of the larger text fragments is out of the scope of these projects.

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 information extraction 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.

Another tool that was tried one a large-scale is SCORE [Sheth2002]. It integrates several information extraction methods, including probabilistic, learning, and knowledge-based techniques, then combines the results from different classifiers.

Much of the work in the information extraction community is aimed at "rule learning", automating the creation of extraction patterns from previously tagged or semi-structured documents [Nobata1999] and unsupervised extraction [Etzioni2005]. This issues our work does not address, however the actual application of the patterns to documents is in many ways similar to our method, in particular, ontology-based method of Embley et als [Wessman2005]. The major differences lie in the implementation whereas Embleys method relies primarily on regular expressions, our approach combines high-speed context-free robust parsing combined with simple word search. Similar to wrappers Embleys approach is intended for processing preferably semi-structured web pages with multiple records: the more structured the page, the better the annotation results. Wrapper induction methods such as Stalker [Muslea2003] and BWI [Freitag2000] which try to infer patterns for marking the start and end points of fields to extract, also relate well to our work. When the learning stage is over

and these methods are applied, their effect is quite similar to our results, identifying complete phrases related to the target concepts. However, our results are achieved in a fundamentally different way by predicting start and end points using phrase parsing in advance rather than phrase induction afterwards. The biggest advantage of wrappers is that they need small amount of training data, but on the other hand they strongly rely on contextual clues and document structure. In this case if the source document would be reorganized, the tool should be retrained on the newly annotated examples. In contrast, our method uses context-independent parsing and does not require any strict input format.

Our approach fundamentally differs from these tools: it uses an extremely lightweight but robust context-free parse in place of tokenization and part-of-speech recognition; our method does not have the learning phase, instead it has to be tuned manually when being ported to a particular application, substituting or extending domain dependent components; it does not necessarily 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. This advantage makes our tool more fast and less dependent on the additional knowledge sources.

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 for bigger inputs, exhibiting very fast and linear performance even without tuning. We consider as contribution of our work also the cost-effective evaluation scheme proposed to measure the quality of annotations.

In future we plan to prove our method for the larger documents, richer conceptual models and different domains. Additionally, we intend to experiment with a number of techniques used in software analysis area that, thus far, we have not taken advantage of: alias resolution, unique naming, architecture patterns, markup refinement and others.

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

[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

[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

[Dean2001] Dean, T., Cordy, J., Schneider, K., Malton, A.: Experience using design recovery techniques to transform legacy systems. In *Proc. 17th Int. Conference on Software Maintenance*, 622-631, 2001

[Etzioni2005] Etzioni, O., Cafarella, M.J., Downey, D., Popescu, A.M., Shaked, T., Soderland, S., Weld, D.S., Yates, A.: Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence* **165**, 91-134, 2005

[Freitag2000] Freitag, D., Kushmerick, N.: Boosted wrapper induction. In *Proc. 17th National Conference on Artificial Intelligence*, 577-583, 2000

[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

[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

[Muslea2003] Muslea, I., Minton, S., Knoblock, C.A.: Active learning with strong and weak views: A case study on wrapper induction. In *Proc. 18th Int. Joint Conference on Artificial Intelligence*, 415-420, 2003

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

[Nobata1999] Nobata, C., Sekine, S.: Towards automatic acquisition of patterns for information extraction. In *Proc. International Conference on Computer Processing of Oriental Languages*, 1999

[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

[Sheth2002] Sheth, A., Bertram, C., Avant, D., Hammond, B., Kochut, K., Warke, Y.: Managing Semantic Content for the Web. *IEEE Internet Computing*, **6(4)**, 80-87, 2002

[Wessman2005] Wessman, A., Liddle, S.W., Embley, D.W.: A generalized framework for an ontology-based data-extraction system. In *Proc. 4th Int. Conference on Information Systems Technology and its Applications*, 239-253, 2005

[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