Managing Document-based Knowledge Lifecycle via Semantic Web Technologies

Ravish Bhagdev, Ajay Chakravarthy, Sam Chapman, Fabio Ciravegna, Aba-Sah Dadzie, Mark Greenwood, Jose' Iria, Vitaveska Lanfranchi and Jonathan Butters.

Department of Computer Science, University of Sheffield,
Regent Court, 211 Portobello Street,
S1 4DP Sheffield, United Kingdom
{N.Surname}@sheffield.ac.uk

Abstract. Max 200 Words

Keywords: application and evaluation of semantic web technologies, ontology-based search, hybrid search, document and knowledge annotation, search and retrieval.

1 Introduction

According to projections from Gartner 2006 [ADD CITE - Forecast: Information Access and Search Technology in the Enterprise, 2006-2010], white-collar workers will spend anywhere from 30 to 40 percent of their time managing documents between 2006-2010; this figure was only 20 percent in 1997. One of the reasons for this is that 80 percent of business is conducted on unstructured information For example about 40,000 unstructured documents are produced every year for documenting a class of jet engines and their instances. As the average expected lifetime of an engine type is around 50-75 years, the number of legacy documents is in the order of several hundred thousands, tending to millions.
The main objective of the challenge project is to build an environment to acquire and share knowledge in large distributed environments in an effective way. Knowledge is acquired from documents (either in the form of legacy documents or while new documents are created) in the form of ontology based annotations. Then knowledge and documents are made available for searching and for qualitative and quantitative analysis of the knowledge and information contained.

Requirements for such an environment are:

- The tool must be able to deploy applications that support the user everyday work and tasks with very limited intrusiveness. For example it must not ask to perform detailed manual annotation of documents, which would distract the users from their tasks. But also, support to management of knowledge must go beyond simple document retrieval.

- It must enable to accommodate different user requirements, tasks and profiles, accommodating, for example different search strategies. This is particularly important considering that some of the technologies used (e.g. automatic document annotation) can be imprecise. Accommodating different strategies implies enabling users to circumvent such limitations to reach the required knowledge.
- It must be portable to different domains and tasks with limited effort and in a limited time.
- It must be able to work over a medium/large scale, coping with dozens (or hundreds) of thousands of documents and thousands (or millions) of triples; documents are expected to be multimedia object containing text and/or images.
- It must be able to integrate knowledge information and documents from distributed heterogeneous sources;
- It must be able to cope with a dynamic world where new information and documents are added constantly and must readapt to such changes.

2 System Architecture

The architecture is fully based on open Web and semantic Web standards. The central means of representation and communication of knowledge are (a set of) ontologies that define both the domain and the user tasks. The architecture is composed of two main parts:

- A set of modules for knowledge acquisition from multimedia user input and legacy multimedia documents. They provide methodologies for unstructured indexing and structured capture of available semantic knowledge. The structured knowledge acquisition uses both modules for manual capture, and automatic discovery of structured knowledge annotations.
- A knowledge sharing module that includes a search module based on hybrid search [1] which enables accommodating different search strategies, from keyword based searches, to ontology-based searching and any combination of the two. It also enables qualitative and quantitative analysis of data and knowledge.

The acquisition architecture is illustrated in figure 1. Documents are either discovered by a crawler or submitted individually to the system. Documents are then classified and the relevant ontologies for annotation identified. Again, this can be done manually by the user or automatically by a classifier. Then the document is annotated using one of the available modules, either automatically (useful for legacy documents) or the information is acquired with the user in the loop during document creation and editing. Available annotation plugins include T-Rex *(cit)* and Saxon (or any of their combination) and AktiveIpas (they are described below)

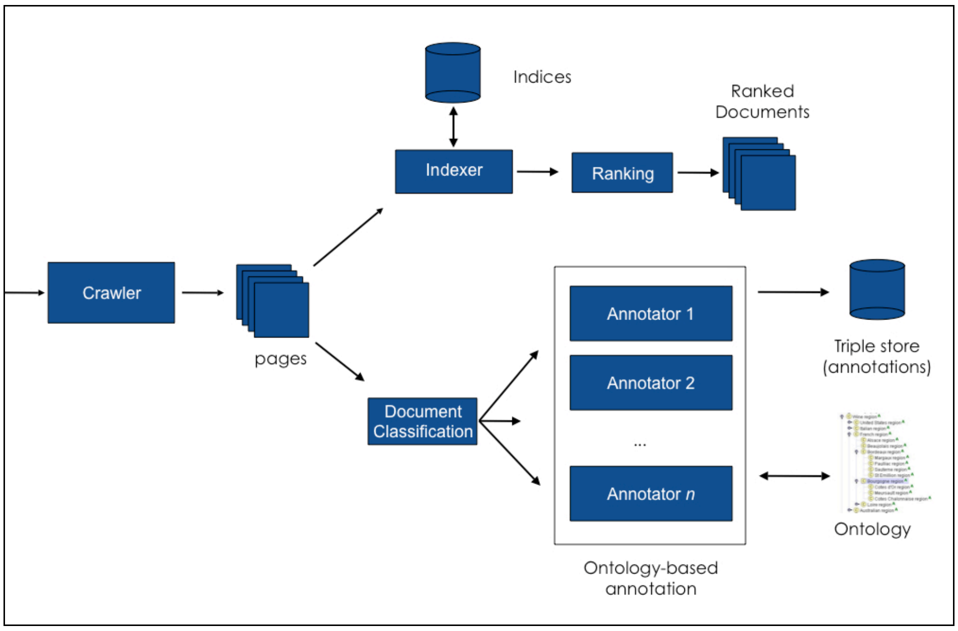

Fig. 1: Document indexing and annotation: traditional keyword indexing and document ranking (top of figure) is done in parallel to ontology-based annotation (bottom).

Extracted information (ontology-based annotations) is stored in the form of RDF triples according to OWL or RDF ontologies into a triple store (currently Sesame or 3Store).

Searching is performed using a Hybrid search strategy that enables mixing keyword-based searches and ontology-based searches [2]. In hybrid search, a query is composed partially by conditions matching the document annotations in the triple store and partially keywords in the document. The organisation of a query execution is shown in Figure 2.

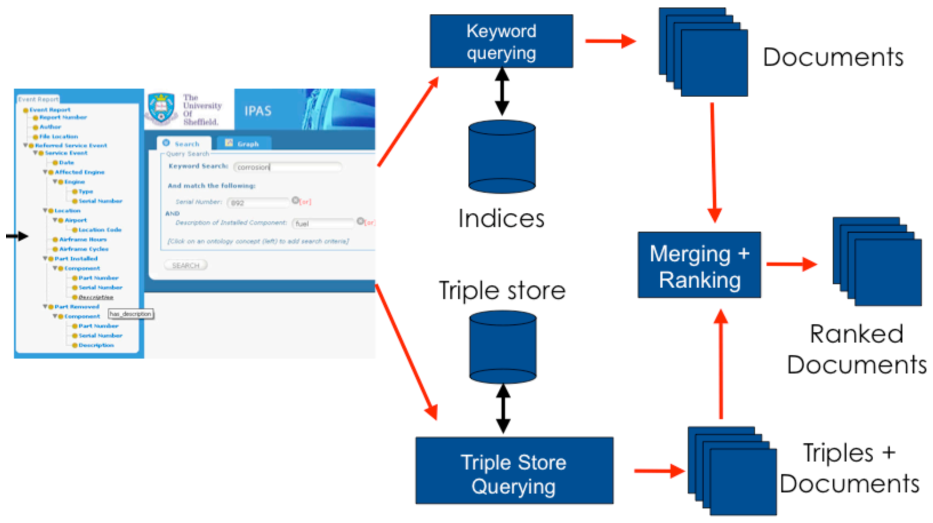

Fig. 2: Combining keywords and ontology-based search in hybrid search.

A user query is composed via an interface that enables easy graphical composition of ontology and keyword-based conditions. The query is then split into the ontology based part (sent to the triple store) and the keywords (sent to the IR system Lucene or the Search Engine Nutch).

Why do we need to mix the two? Motivations from ISWC paper (Unfortunately, IE is a technology that performs very well on simple tasks (such as named entity recognition), but poorly on more complex tasks such as event capture [3, 4]. Therefore, sometimes, automatic annotation is unsuitable, at least for some parts of an ontology. When manual annotation for these parts is unfeasible, some of the metadata is unavailable. …)

In the rest of the section we will focus on the three main modules: two for annotation and one for search.

2.1 Editing and Annotating Document

AKTiveForm is a tool to enabled controlled domain specific document writing. The idea starts from noticing that in large organisations many documents are produced as output of a specific process, but very often are generated as text documents (e.g. Word files) containing semi-structured portions (e.g. forms and tables), images and free text parts. Tables are seldom standardised and so are forms which change in time when authors decide so, because the Word format enables that. AktiveForm enables the easy definition of forms and tables in documents so to constraint the user activities. Form generation is controlled by an ontology that enables declaring what forms are to be generated, their layout and the constraints on the single fields. This is done using an OWL ontology (defining relations among fields such as precedence and layout) and a reasoner to create the actual realisation of the form and the document (implemented in AJAX). AktiveForm enables the input of data and free text, the checking of the value inserted in fields (e.g. for example checking that invalid values are inserted) and enabling automatic annotation with semantic tags of the values. A terminology recogniser is associated (via ontology) to a form fields in order to help users identifying fixed sets of values in large sets. For example, there are about 300,000 jet engine components; in order to input the name of one of them, the user can write a simple natural language description and then ask for help. The relevant terms are then presented. Terminology recognition is here applied using a mixture of context and string distance metrics [3].

It is possible to add free text in some fields and annotating it using technologies similar to the one we implemented in AktiveMedia [4]. Images can be added and annotated. The system can use an underlying information extraction system (T-Rex, see below) to reduce the burden of annotation, as also done in Aktivemedia.

When the document has been created filling all parts of the form, a PDF file is generated containing the whole generated document.

All the fields with semantic annotations can then be passed directly to the search capabilities of X-Search. The interesting aspects of AktiveForm are:

- the generation of application is completely declarative and done via ontology and reasoning; changes do not require any programming; just the modification of a simple ontology.
- the system works in a browser on a client server architecture; either in local or with a remote server. Therefore it can be used even in mobile situations (quite frequent for example in garages or hangars) and synchronised in a second time.
- RDF-based ontology-based annotation is generated as side effect of filling fields of a form. For free text parts annotation is either inputted by the user using a simple intuitive colour coded interface (see Figure Y) with the support of the IE system;
- The insertion of AktiveForm (and its semantic capabilities) in a work environment does not require any change in terms of workflow for the users. They can still generate apparently the same document as before; the only difference is that instead of filling an unstructured text document, now they fill HTML forms that produce as a side effect RDF data. The HTML form can appear nearly identical to the Word file.
- At the end of the process a PDF file is generated so that it can be printed or sent by email; we do not generate Microsoft Word to avoid modifications to the generated document. The only way to modify it is to modify the content of the forms in order to maintain consistency of information.

2.2 Annotating Legacy Document

Put intro here about the importance of capturing legacy documents and the impossibility to annotate them manually – from ISWV paper

TRex and Saxon description

Insert here more information about T-Rex from the architectural point of view (SVM, applications, etc.)
Also explain that does name recognition/relation extraction, ontology population and table field extraction (maybe mention that this is required according to the user req)
Explain what are the good points about TRex

TRex uses an internal formalism called RuneStone [LREC paper] which is shared by Saxon, so that modules and results built with TRex and Saxon can be interleaved in an architecture.

Saxon is a rule based tool for annotating documents. Saxon rules are regular expressions over document features, such as tokens and sentences. The output of a rule can be either other annotations or unrestricted actions specified within the rule.

Saxon assumes the document is represented as a graph, with nodes representing document sections (tokens, sentences, etc.) and edges representing relationships between sections (this token occurs in this sentence, this token follows this token, etc.). Saxon rules are then defined as regular expressions detailing how to move between sections of the document. A rule has three main parts a starting point, a regular expression describing how to move between sections of the document, and a section detailing how the document should be updated if the rule matches. For example a simple rule to match the names of people can be defined as:

```
    Rule:Person
  Token{token_string{~Mr|Dr|Mrs|Miss}}
  (token_next{token_pos{=NNP}})+
```

This defines a rule called person which will start matching at any token in a document which represents one of the four titles Mr, Dr, Mrs, or Miss. The last line of the rule specifies how matching should progress from this initial starting point. Assuming a token representing a title has been found then the rule matches one or more following tokens with a part-of-speech tag which identifies the token as a proper noun. This rule does not specify any action to take when the rule matches and so a simple annotation is added to the document specifying that a Person (i.e. the rule name) consisting of the matched tokens has been found.

As noted the example rule is very simple and does not fully exploit the features of Saxon. The flexibility of Saxon stems from the ability to specify unrestricted Java code as the action to take when a rule matches allowing any possible modification of the document to be performed. For more complex rule examples which fully exploit Saxon see the documentation available at http://nlp.shef.ac.uk/wig/tools/saxon/.

Saxon places no restriction on how a document should be represented; allowing different document formats to be represented in the most appropriate way. Rules can then be developed to annotate any given document representation. This flexibility makes Saxon an ideal annotation tool, which has currently been used to implement both a gazetteer (containing approximately 80,000 terms) and a simple named entity tagger.

2.3 Searching Knowledge and Documents

How do we search and why? How do we support sharing? Why is it worth? What are the issues with other types of search that makes it worth having x-search?

At retrieval time, hybrid search requires the following steps:

- the query is parsed and the three types of searches identified (keywords, keywords-in-context and ontology-based) and separated;
  - keywords are sent to the traditional information retrieval system; this will return the identifiers (URIs) of all the documents containing those keywords; standard tools perform two types of matches: strict matches, where all keywords must be present in the returned documents (this is what most company search tools do) or less strict matches where some of the keywords can be missing from the documents (search engines tend to do this); X-Search uses Nutch for indexing documents. The reason for using Nutch is its high quality keyword mechanism and its ability to exploit all the strategies for ranking used by search engines.
  - queries about concepts (and their relations) are matched with the facts in the knowledge base using a query language like SPARQL[1]; results can be returned that strictly match the results; in a more sophisticated approach it is possible to perform near matches, for example by automatically relaxing constraints; Concerning support for triple stores, X-Search provides plugins for Sesame and 3store; query languages supported are SPARQL and Sesame's SeRQL.
  - queries of keywords-in-context are sent to the knowledge base, returning conceptual instances containing the given keywords (again using SPARQL); again near matches can be performed;
- Finally, the results of the different queries are merged, ranked and displayed. These are discussed below.

As multiple constraints can be added to the query, the **logical language** is restricted in order to provide a simple and intuitive interface. Only some very common Boolean combinations are supported for querying. This decision was supported by the observation that in carrying out their tasks, users adopted strategies that do not require the full logical language; furthermore research done in human-computer interaction shows that graphical representation of the whole Boolean logic is not understood by users [10,11].

[1] www.w3.org/TR/rdf-sparql-query/

AND constructs are allowed among conditions checking different concepts in the ontology. So for example, *contains*(*removed-component, "fuel") AND contains(jet-engine-name, "Trent"*) is acceptable, but *contains*(*removed-component, "fuel") AND contains(part removed, "meter")* is not. The latter is acceptable if formulated as *contains* (*removed-component, "fuel meter")*. Conditions in AND are displayed on different lines in the interface (Fig. 3 shows an example of a combination of *removed-component AND operational-effect*).

OR constructs are acceptable only if between conditions on the same concept. So contains(removed-component, "fuel") OR contains(removed-component, "meter") is accepted, but contains(removed-component, "fuel") OR contains(jet-engine-name, "Trent") is not. The latter must be split into two different queries.

Merging of results. A direct matching between keyword and ontology-based results is not straightforward as their results are incompatible. Keyword matching returns an <u>ordered</u> set of *URIs* of documents (*uriOrdSet*) of size *n*.
A semantic repository *R* is instead queried according to an ontology: such a query returns an <u>unordered</u> set *rSet* (size *m*) of individual assertions < *subj, rel, obj*>[2]

Using the provenance information associated to each triple, it is possible to compute the set of documents that contain the required information.

The list of *URIs* of documents generated using provenance information is now directly compatible with the output of keyword matching. The result of the query is given by the intersection of the two sets of document *URIs*.

Ranking. As shown by a number of studies, proper ranking (i.e. the ability to return relevant documents first) is extremely important for a positive user experience. The results returned by the different modalities provide material for orthogonal ranking methods:

- keyword based indexing systems like Nutch enable ranking of documents according to (1) their ability to match the keyword-based query; (2) the keywords used in anchor links (i.e. the text associated to hyperlinks pointing to a specific document) and (3) the document popularity measured as function of the weight of the links referring to the document itself.
- ontology-based search ranks according to the presence and quality of metadata.

Currently ranking is performed by relying on the Nutch ranking. This is because – as explained above – Nutch's ranking is very reliable and uses a number of strategies, including hyperlinking and anchor text matching. Moreover, as the matching on the ontology part of the query is strict (i.e. only the documents that match <u>all</u> the conditions are returned), all the documents tend to be equivalent in content. However, the visualization interface enables the user to change the ranking by focusing on specific metadata values.

Visualization. Results can be presented according to a number of dimensions: as a list of ranked documents, as aggregated metadata (e.g. via graphs) with associated provenance, etc. Again there is an incompatibility here between the results of

[2] Both ontology-based and keyword in context queries are covered here.

ontology-based search (where it is possible to aggregate metadata), and keyword-based search where it is possible only to count words or returned documents.

The resulting interface works in a standard Web browser, is form-based and enables the definition complex hybrid queries in an intuitive way. Keywords can be inserted into a default form field in a way similar to that required by search engines; Boolean operators AND and OR can be used in their combination. Conditions on the metadata can be added to the query by clicking on the ontology graph. This creates a form item to insert conditions on the specific concept.

The returned set of documents is displayed as a list on the mid-right panel of the interface (see fig. 4); each item in the list is identified by the title (or file name) of the document and the values in the metadata that satisfy the ontology-based search. Clicking on one item in the list causes the corresponding document to be shown on the bottom right. The document is presented in its original layout with added annotations via colour highlighting; advanced features or services are associated to annotations [12, 13]: for example right clicking on a concept enables – among other things - query expansion with the selected term. Multiple documents can be opened simultaneously in different tabs.

3.1 Quantitative Analysis of information (graphs)

One of the identified user requirements is to support quantitative analysis of the retrieved data by automatically generating graphs and charts. X-Search allows user to create bi-dimensional graphs by choosing the style (pie or bar chart) and the variables to plot. The graph in Figure 4 plots the results of the previous query by location and engine type. Each graphic item (each bar in the example) is active and can be clicked to focus on the sub-set of documents that contains that specific occurrence.

4 Application and Experiments

We applied the tool to two classes of applications:

- Company's knowledge management; application done for Rolls-Royce plc; two corpora containing dozens of thousands of documents (Event reports written by airline representatives on issues on jet engines and Technical Variance requests, again written by airline representatives to ask for variation to the specifications of an engine and including the official answer provided by the jet engine company). A third testbed for writing Module Condition Report in overhaul facilities is under development.
- Historical information search in distributed archives for London of the18th century. This is a more limited application, which enables historians to navigate data about London 18th Century by simultaneously searching data in multiple archives.

Tests were carried out to evaluate the effectiveness and the user acceptance of the hybrid search paradigm. Tests were designed to generalise over the use of the specific implementation of hybrid search, with its specific query formalism and interface, with specific strategies for visualisation, indexing, etc. For example for the aerospace domain, evaluation was performed in two ways:

- *in vitro*: queries generated from real work tasks were issued using three options: keyword-based searching, ontology-based searching and hybrid searching; this test enabled us to evaluate the effectiveness of the method in principle;
- *in vivo:* 32 Rolls-Royce plc employees and 5 historians were involved in a usability test of the applications and commented on a number of aspects such as efficiency, effectiveness, etc.; this evaluation enabled measuring the extent to which users understand the hybrid search paradigm and feel that it returns appropriate results.

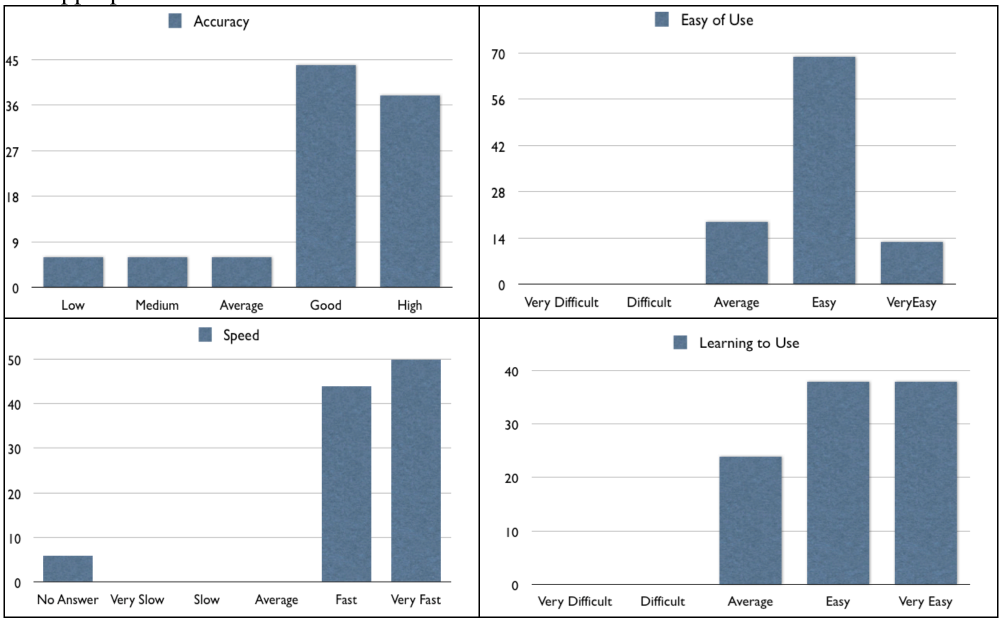

Fig. 7: Results of evaluation of X-Search by 32 users (values are in %).

Results of tests were published in [1] and show that the accurate design of contextual acquisition of acquisition and sharing/reuse brings very good results with high acceptance by users. The next figure shows the user acceptance of the sharing/searching methodology as registered assessing the validity of the hybrid search paradigm as well as the usability of the X-Search system (Fig. 7):

- Use of hybrid search: all users appeared to have grasped the concept of hybrid search. We noticed that users adopted different strategies: some used first keyword-based search and added conditions on the ontology in a second iteration; others instead composed conditions on ontology and keywords in a single search; others used ontology-based search as first approach and added keywords later to refine the task. This means that different approaches to searching can be accommodated in the hybrid search framework.

- Learnability: How easy is to learn to use the hybrid approach: 75% of users found easy or very easy to learn the system. 25% said it was average.
- System accuracy: system reliability in retrieving relevant documents; was high with 82% judging X-Search reliable or highly reliable; although this could seem a feature of the system rather than of hybrid search, in our view the comment refers to the fact that with hybrid search the searches were effective.
- Experience in searching: 82% of users found X-Search easy or very easy to use; the ease of use was a concept often commented about in the interview;
- System Speed: the system was judged fast or very fast in executing the query allowing a quick task completion by 98% of users.

5 Conclusions and Future Work

- why is this interesting?

- what are we demonstrating?

- why should people care about us?

Currently the application to event reports is in beta test at Rolls Royce plc with the involvement of hundreds of users. The TV application will be released in September 2007. The historian application is in alpha test at the university of Hertfordshire.

Acknowledgments. We would like to thank Colin Cadas (Rolls-Royce) for the constant support in the past two years. The work was jointly supported by IPAS, a project funded by the UK DTI (Ref. TP/2/IC/6/I/10292) and Rolls-Royce plc, X-Media, (www.x-media-project.org) sponsored by the European Commission as part of the Information Society Technologies (IST) programme under EC grant number IST-FP6-026978. The application to the history domain, was sponsored by the Arts and Humanities Research Council under the contract Armadillo. Thanks to Matthew Rowe, Lei Xia for helping in the preparation of this challenge.

References

1. Chakravarthy, A., Lanfranchi, V., Ciravegna, F.: Cross-media Document Annotation and Enrichment, Proceedings of the 1st Semantic Authoring and Annotation Workshop, 5th International Semantic Web Conference (ISWC2006), Athens, GA, USA, 2006
2. McCallum, A.: Information Extraction: Distilling Structured Data from Unstructured Text, ACM Queue, Vol. 3 No. 9 - November 2005.
3. Marsh, E., Perzanowski, D.: MUC-7 Evaluation of IE Technology: Overview of Results, Proceedings of the 7th Message Understanding Conference Proceedings, http://www-nlpir.nist.gov/related_projects/muc/proceedings/muc_7_toc.html
4. Ireson, N., Ciravegna, F., Califf, M.E., Freitag, D., Kushmerick, N., Lavelli, A.: Evaluating Machine Learning for Information Extraction, Proceedings of the 22nd International Conference on Machine Learning (ICML 2005), Bonn, Germany, 2005
5. Kiryakov, A., Popov, P., Terziev, I., Manov, D., Ognyanoff, D.: Semantic annotation, indexing, and retrieval, Journal of Web Semantics, Vol 2 (1), 49-79

6. Gilardoni, L., Biasuzzi, C., Ferraro, M., Fonti, R., Slavazza, P.: LKMS – A Legal Knowledge Management System exploiting Semantic Web technologies, Proceedings of the 4th International Conference on the Semantic Web (ISWC), Galway, November 2005.
7. Uren, V. S., Cimiano, P., Iria, J., Handschuh, S., Vargas-Vera, M., Motta, E., Ciravegna, F.: Semantic annotation for knowledge management: Requirements and a survey of the state of the art. Journal of Web Seamantics, Volume 4 (1), 14-28, 2006
8. Dill, S., Eiron, N., Gibson, D., Gruhl, D., Guha, R.V., Jhingran, A., Kanungo, T., McCurley, K.S., Rajagopalan, S., Tomkins, A. , Tomlin, J.A., Zien, J. Z.: A case for automated large-scale semantic annotation. Journal of Web Semantics, Volume 1(1), 115-132, 2003
9. Iria, J., Ireson, N., Ciravegna, F.: An Experimental Study on Boundary Classification Algorithms for Information Extraction using SVM. In Proceeding of the EACL 2006 Workshop on Adaptive Text Extraction and Mining (ATEM 2006), at 11th Conference of the European Chapter of the Association for Computational Linguistics, April 2006.
10. Shneiderman, B.: Designing the User Interface (3rd edition). Addison-Wesley, 1997.
11. Hertzum M., Frokjaer, E.: Browsing and querying in online documentation: a study of user interfaces and the interaction process. ACM Transactions on Computer-Human Interaction 3(2):136–161, 1996.
12. Dzbor, M. - Domingue, J. B. - Motta, E.: Magpie - towards a semantic web browser. 2nd Intlernational Semantic Web Conference (ISWC), Sanibel Island, Florida, USA, 2003.
13 Lanfranchi, V., Ciravegna, F., Petrelli, D.: Semantic Web-based Document: Editing and Browsing in AktiveDoc, Proceedings of the 2nd European Semantic Web Conference , Heraklion, Greece, 2005.