Integrating Keywords and Semantics on Document Annotation and Search

Nikos Bikakis[1,2], Giorgos Giannopoulos[1,2], Theodore Dalamagas[2] and Timos Sellis[1,2]

[1] Knowledge & Database Systems Lab | National Technical University of Athens | Greece

[2] Institute for the Management of Information Systems | "Athena" Research Center | Greece

bikakis@dblab.ntua.gr • giann@dblab.ntua.gr
dalamag@imis.athena-innovation.gr • timos@imis.athena-innovation.gr

Abstract. This paper describes GoNTogle, a framework for document annotation and retrieval, built on top of Semantic Web and IR technologies. GoNTogle supports ontology-based annotation for documents of several formats, in a fully collaborative environment. It provides both manual and automatic annotation mechanisms. Automatic annotation is based on a learning method that exploits user annotation history and textual information to automatically suggest annotations for new documents. GoNTogle also provides search facilities beyond the traditional keyword-based search. A flexible combination of keyword-based and semantic-based search over documents is proposed in conjunction with advanced ontology-based search operations. The proposed methods are implemented in a fully functional tool and their effectiveness is experimentally validated.

Keywords: *GoNTogle, Semantic Annotation, Document Annotations, Ontology based Retrieval, Hybrid Search, Semantic Search, Keyword Search.*

1 Introduction

Document annotation and search have received tremendous attention by the Semantic Web [2] and the Digital Libraries [3] communities. Semantic annotation involves tagging documents with concepts (e.g., ontology classes) so that content becomes meaningful. Annotations help users to easily organize their documents. Also, they can help in providing better search facilities: users can search for information not only using keywords, but also using well-defined general concepts that describe the domain of their information need.

Although traditional Information Retrieval (IR) techniques are well-established, they are not effective when problems of concept ambiguity or synonymity appear. On the other hand, neither search based only on semantic information may be effective, since: a) it does not take into account the actual document content, b) semantic information may not be available for all documents and c) semantic annotations may cover only a few parts of the document.

Hybrid solutions that combine keyword-based with semantic-based search deal with the above problems. Developing methodologies and tools that integrate document annotation and search is of high importance. For example, researchers need to be able to organize, categorize and search scientific material (e.g., papers) in an efficient and effective way. Similarly, a press clipping department needs to track news documents, annotating specific important topics and searching for information.

This paper describes GoNTogle, a framework for document annotation and retrieval, built on top of Semantic Web and IR technologies. GoNTogle provides both manual and automatic ontology-based annotations, supporting documents of several formats (doc, pdf, txt, rtf, odt, sxw, etc.). Annotation is based on standard Semantic Web technologies like, OWL and RDF/S. All annotations are stored in a centralized server, providing a collaborative environment. A learning method, exploiting textual information and user annotation history, is proposed to support the automatic annotation mechanism.

GoNTogle also provides three search types: a) *Keyword-based*, b) *Semantic-based* and c) *Hybrid*. Experimental evaluation validates the effectiveness of the proposed hybrid method, compared to the other two. Finally, several advanced ontology-based searching operations are provided, including the capability to expand or shrink the result list using ontology information, in order to retrieve higher quality results.

Regarding the design principles of our framework, they are based on the requirements set in previous works [4, 5, 7]. In contrast with the existing approaches, our aim was to design an easy-to-use document annotation and search framework that supports (a) viewing and annotating popular document types while maintaining their initial format, (b) offering a collaborative environment by sharing those annotations (c) supporting Semantic Web standards, (d) integrating textual information with semantics and (e) supporting a flexible combination of keyword-based and semantic-based search in conjunction with advanced ontology-base search operations.

Contributions. The main contributions of this work are summarized as follows:

1. We have designed and implemented an easy-to-use document annotation framework that supports the most widely used document formats, providing also advanced search facilities.
2. The framework is based on a server-based architecture, where document annotations are stored in a central repository, separately from the original document. This offers a collaborative environment where users can annotate and search documents.
3. We propose a learning method for automatic annotation of documents based on models trained from user annotation history and textual information, so that annotation suggestions are tailored to user behavior.
4. We introduce a hybrid search method that provides a flexible combination of traditional keyword-based and semantic-based search for effective document retrieval.
5. We present a user-based evaluation to demonstrate the effectiveness of the automatic annotation method. Moreover, we demonstrate a comparative evaluation to validate that the proposed hybrid search outperforms keyword-based and semantic-based search in terms of precision and recall.

Paper Outline. The rest of the paper is organized as follows. The semantic annotation mechanism is presented in Section 2, while Section 3 describes the search facilities. Section 4 presents the system architecture and provides technical information about the implementation. Section 5 presents the evaluation of our proposed methods. Section 6 discusses the related work and, finally, Section 7 concludes our work.

2 Semantic Annotation

GoNTogle framework supports semantic, ontology-based annotations, for widely used document formats (doc, pdf, txt, rtf, odt, sxw, etc.). It allows annotating the whole document or parts of it. GoNTogle framework supports both manual and automatic annotations. For automatic annotation we propose a learning method that exploits user annotation history and textual information to automatically suggest annotations for new incoming documents.

GoNTogle provides a common ontology-based annotation model (Figure 1) for all supported document formats. Annotations are stored on a centralized ontology server, separately from the original document. Annotations from different document formats are defined and stored in exactly the same way. Each annotation is stored as an ontology class instance, along with information about the annotated document. We define a set of ontology properties that are used to store the minimum essential information needed to provide a bidirectional connection between documents and ontologies. These properties contain information like: document URL, annotation offsets, page number, extent of annotation over the document, etc.

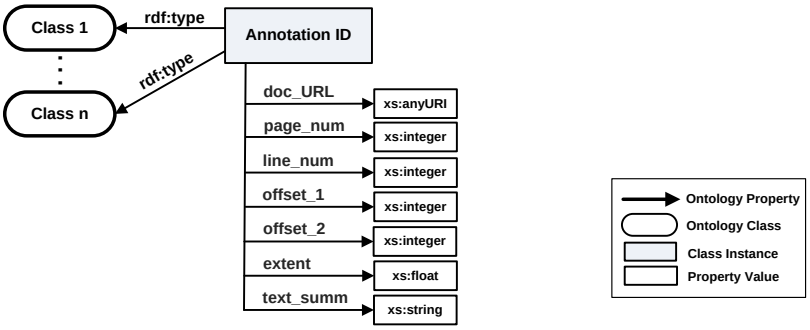

Figure 1. Ontology-based annotation model

Figure 1 shows the ontology-based annotation model we developed in the context of the GoNTogle framework. Annotations are represented as class instances that can belong to one or more ontology classes. Using ontology properties, all the essential annotation information is attached to these instances. Property *doc_URL*, corresponds to the document's URL (including document's file name) of represented annotation. *page_num* and *line_num* properties, correspond to the number of the page and line respectively where the annotation begins. The property *offset_1* corresponds to a number that indicates the offset from the beginning of the document until the

beginning of the annotation. As the same, property *offset_2* corresponds to the offset from the end of annotation until the end of the document. The property *extent* represents the extent of the annotation over the document. Finally, *text_summ* used for storing the summary of the annotated text (i.e., 1-3 tokens from the begin and the end) required for the GUI functionality.

2.1 Automatic Semantic Annotation

In this section, we present the learning method used for automatic document annotation. We propose a method based on *weighted kNN* classification [1] that exploits user annotation history and textual information to automatically suggest annotations for new documents. Next, we describe our approach in detail.

The training data of our method include document annotations provided manually by the users. When a document is manually annotated, the annotation text is extracted and indexed using an inverted index. Along with the textual information, the index also stores information about the annotation classes for each annotated document (or part of document).

Annotation Suggestion Algorithm

Input: *selected text st, index I*

Output: *suggested class* cl_i, *suggested class score* Scr_{cli}

1. **for each** annotated text *at* **in** *I*
2. **calculate** $ts_{st,at}$
3. **end for**
4. **Insert** the *k* most similar annotated texts in *S*
5. **for each** *at* **in** *S*
6. **for each** class *cl* annotate *at*
7. $Scr_{cl} = Scr_{cl} + (w_1 * ts_{st,at}) * (w_2 * e_{cl,at})$
8. **end for**
9. **end for**
10. **return** cl_i, Scr_{cli}

Figure 2. Annotation suggestion algorithm

To automatically annotate documents, the user first selects a document or a part of it. Then, given the set of training data, our method suggests a ranked list of ontology concepts (classes) to annotate the document (or its part). Figure 2 presents our method. It takes as input the selected text *st* and the inverted index *I*. Based on textual similarity $ts_{st,at}$ between *st* and each indexed annotated text *at*, the *k* most similar annotated texts are considered for further processing, and included in set *S* (*lines 1-4*). Then for each *at* in *S*, we retrieve the ontology classes used to annotate *at*. Each class *cl* is given a score Scr_{cl} that combines (a) the textual similarity (based on Lucene similarity model[1]) score $ts_{st,at}$ between *st* and *at* and (b) a score $e_{cl,at}$ representing the *extent* to which each *at* in *S* is annotated with class *cl* (*line 7*). As $e_{cl,at}$

[1] http://lucene.apache.org/java/3_0_1/api/core/org/apache/lucene/search/Similarity.html

we define, the number of tokens of the *cl* annotations in *at* divided by the number of tokens in *at*.

$$e_{cl,at} = \frac{number\ of\ tokens\ \ of\ cl\ annotations\ over\ at}{number\ of\ tokens\ in\ at}$$

The w_1 and w_2 weights are used to quantify the preference of textual similarity against semantic similarity (or vice versa). Finally, a ranked list of suggested annotation classes cl_i and their score Scr_{cli} is presented to the user (*line 10*). The user may choose one or more suggested classes to conclude the automatic annotation process.

3 Search

In this section, we present the search facilities proposed in the context of GoNTogle framework. We formally define the supported search types (Section 3.1) and we analyze the ontology-based advanced search operations (Section 3.2). Moreover, we introduce the hybrid search method, which combines keyword-based and semantic-based search. Below we introduce the notation used in the following paragraphs.

Symbol	Notation
q_{key}	Keyword query, consisting of search term $\{t_1, t_2,\ldots t_m\}$
$S_{key}(q_{key})$	Keyword-based search
RS_{key}	Keyword-based search result set
$Scr_{key}(q_{key},d)$	Keyword-based similarity score
q_{sem}	Semantic query, consisting of search classes $\{cl_1, cl_2,\ldots cl_n\}$
$S_{sem}(q_{sem})$	Semantic-based search
RS_{sem}	Semantic-based search result set
$Scr_{sem}(q_{sem},d)$	Semantic-based similarity score
$S_{hybr}(q_{sem},q_{key})$	Hybrid search
RS_{hybr}	Hybrid search result set
$Scr_{hybr}(q_{sem},q_{key},d)$	Hybrid similarity score

3.1 Search Types

We categorize the basic search facilities of our framework into three types: a) *Keyword-based* search*, b)Semantic-based* search *and c) Hybrid* search*.*

Keyword-based search. This is the traditional search model. The user provides keywords and the system retrieves relevant documents based on textual similarity. We adopted the text similarity metric used in Lucene IR engine.

Keyword-based search is denoted as $S_{key}(q_{key})$, where $q_{key}=\{t_1, t_2,\ldots t_m\}$ and t_i are the search terms with $m\geq 1$.

Keyword-based search returns an ordered *Result Set* RS_{key} of tuples $<d, Scr_{key}(q_{key},d)>$, containing all the documents d matched with terms q_{key}. $Scr_{key}(q_{key},d)$ is the similarity score of document d for the searching terms q_{key}. This score is based on document textual similarity with the searching terms.

Semantic-based search. This search facility allows the user to navigate through the classes of an ontology and focus their search on one or more of them.

Semantic-based search is *denoted as* $S_{sem}(q_{sem})$, where $q_{sem}=\{cl_1, cl_2, \ldots cl_n\}$ and cl_i are the searching classes with $n \geq 1$.

It return an ordered *Result Set* RS_{sem} of tuples $<d, Scr_{sem}(q_{sem},d)>$, containing all the documents d that have been annotated with one or more of the search classes q_{sem}. $Scr_{sem}(q_{sem},d)$ is the similarity score of document d for the searching classes q_{sem}. This score is based on semantic similarity between the searching classes q_{sem} and document d. To define semantic similarity $ss_{cli,d}$ between a class cl_i and a document d, we consider the *extent* of the class annotations over the document: that is the number of tokens used to define the class annotations in d divided by the number of tokens in d.

The final similarity score is defined as follows:

$$Scr_{sem}(q_{sem}, d) = \sum_{i=1}^{n} ss_{cl_i,d} / n$$

$$ss_{cl_i,d} = \frac{number\ of\ tokens\ \ of\ cl_i\ annotations\ over\ d}{number\ of\ tokens\ in\ d}$$

where n is the number of ontology classes used during the semantic-based search, and $ss_{cli,d}$ is a score representing the *extent* to which document d *i*s annotated with class cl_i.

Hybrid search. The user may search for documents using keywords and ontology classes. She can, also, determine whether the results of her search will be the intersection or the union of the two searches.

Hybrid search is denoted as $S_{hybr}(q_{sem},q_{key})=S_{sem}(q_{sem})\ Op\ S_{key}(q_{key})$, where $q_{sem}=\{cl_1, cl_2, \ldots cl_n\}$ and cl_i are the searching classes with $n \geq 1$, $q_{key}=\{t_1, t_2, \ldots t_m\}$ and t_i are the searching terms with $m \geq 1$ and *Op* the Boolean operators *OR* or *AND*.

Hybrid search returns an ordered *Result Set* RS_{hybr} of tuples $<d, Scr_{hybr}(q_{sem},q_{key},d)>$, the contents and the order of the result set depend on *Op* value:

- *Op=AND.* The *Result Set* contains all the documents d that have been annotated with one or more of the search classes q_{sem} and match with terms q_{key}.

  $$RS_{hybr} = RS_{key} \bigcap_{over\ d} RS_{sem}$$

  The final similarity score is defined as:

  $$Scr_{hybr}(q_{sem}, q_{key}, d) = Scr_{sem}(q_{sem}, d) * w_3 + Scr_{key}(q_{key}, d) * w_4$$

  where $Scr_{sem}(q_{sem},d)$ is the similarity score from semantic-based search, and $Scr_{key}(q_{key},d)$ is the similarity score from keyword-based search. The w_3 and w_4 weights are used to quantify the relative importance of the semantic-based and keyword-based scores, when both keyword and semantic queries must be satisfied.

– *Op=OR*. The *Result Set* contains all the documents *d* that have been annotated with one or more of the searching classes q_{sem} and all the documents *d* matched with terms q_{key}.

$$RS_{hybr} = RS_{key} \bigcup RS_{sem}$$

The final similarity score is defined as:

$$Scr_{hybr}(q_{sem}, q_{key}, d) = Scr_{sem}(q_{sem}, d) * w_5 + Scr_{key}(q_{key}, d) * w_6$$

where $Scr_{sem}(q_{sem},d)$ is the similarity score from semantic-based search, and $Scr_{key}(q_{key},d)$ is the similarity score from keyword-based search. The w_5 and w_6 weights are used to quantify the relative importance of the semantic-based and keyword-based scores, when either keyword or semantic queries must be satisfied.

3.2 Advanced Search Operations

Here we present a set of advanced search operations that can be used after an initial search has been completed.

Find related documents. Starting from a result document *d*, the user may search for all documents that have been annotated with a class *cl* that also annotates *d*. For example, if a user had initially searched with class *H.2[DATABASE MANAGEMENT]* [2] and selected one of the results that is also annotated with class *H.2.5[Heterogeneous Databases]*, then "*Find related documents*" would return all documents annotated with both classes.

Find similar documents. This is a variation of the previous search facility. Starting from a result document *d*, the user may search for all documents that are already in the result list and have been annotated with a class *cl* that also annotates *d*. For example, if a user had initially searched with keyword "*XML*" AND class *H.2[DATABASE MANAGEMENT]* and selected one of the results that is also annotated with class *H.2.5[Heterogeneous Databases]*, then "*Find similar documents*" would return all documents annotated with both classes and contained the keyword "*XML*".

Get Next Generation. The resulting list from a semantic-based (or hybrid) search can be confined by propagating the search on lower levels in the ontology (i.e., if class *cl* has been used, then search is propagated only in direct subclasses of *cl*). This is the case when the search topic is too general. For example, if a user had initially searched with *H.2[DATABASE MANAGEMENT]*, then "*Get Next Generation*" would return all documents annotated with at least one of its subclasses (*H.2.5[Heterogeneous Databases], H.2.3[Languages],* etc.).

Get Previous Generation. This offers the inverse functionality of the previous option. The resulting list from a semantic-based (or hybrid) search can be expanded by propagating the search on higher levels in the ontology (i.e., if class *cl* has been used, then search is propagated only in direct superclasses of *cl*). This is the case

[2] We turned the ACM Computing Classification (http://www.acm.org/about/class/) into an OWL ontology.

when a search topic is too narrow. For example, if a user had initially searched with *H.2[DATABASE MANAGEMENT]*, then "*Get Previous Generation*" would return all documents annotated with its superclass (*H.[Information Systems]*).

Proximity Search. This search option allows the user to search for documents that belong to all subclasses of a selected class, by applying a ranking model based on ontology hierarchy. That is, if class *cl* is the initial class, then search is propagated in all direct and indirect subclasses of *cl*. The resulting documents gathered from all levels of the ontology hierarchy are weighted properly (i.e., documents from the selected class *cl* get higher score than 1st level subclasses and even higher than 2nd level subclasses).

4 System Overview

4.1 System Architecture

Due to its centralized server-based annotation storage and management architecture, GoNTogle offers a collaborative user environment. Annotations are stored separately from the original document and may be shared by several user groups. GoNTogle's architecture is presented in Figure 3. The system is divided into 4 basic components:

a) *Semantic Annotation Component* provides facilities regarding the semantic annotation of documents. It consists of 3 modules: (i) Document Viewer, (ii) Ontology Viewer and (iii) Annotation Editor.
b) *Ontology Server Component* stores the semantic annotations of documents in the form of class instances. It consists of 2 modules: (i) an Ontology Manager and (ii) an Ontology Knowledge Base.
c) *Indexing Component* is responsible for indexing the documents using inverted indexes.
d) *Search Component* allows users to search for documents using a flexible combination of textual (keyword-based search) and ontology (semantic-based search) information.

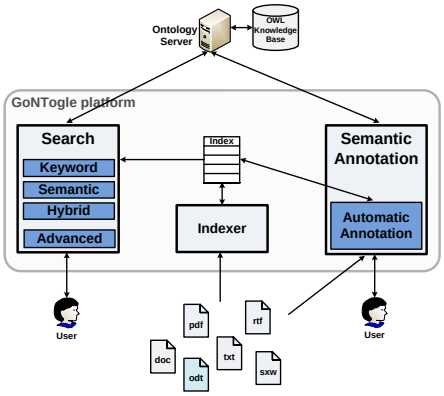

Figure 3. GoNTogle architecture

4.2 Semantic Annotation In-Use

Semantic Annotation Component offers 2 primary functionalities: (a) annotation of whole document and (b) annotation of parts of a document. Also, a user may choose between manual and automatic annotation.

Figure 4 shows the Semantic Annotation window of our application. The user may open a document in the Document Viewer, maintaining its original format. Moreover, she can load and view the hierarchy of an ontology through the Ontology Viewer. In the specific example, the loaded ontology corresponds to the ACM Computing Classification hierarchy. The user can, then, select one or more ontology classes and manually annotate the whole document or part of it. The annotation is stored as an ontology class instance in the Ontology Server, along with information about the annotated document. At the same time, an annotation instance is added in the Annotation Editor list. Each record of this list corresponds to an annotation stored in the Ontology Server. For example (Figure 4), the abstract of the document is annotated with class *H.2.3 [Languages]:Query Languages*, while the whole document is annotated with class *H.2[DATABASE MANAGEMENT]*. The user can manage those annotation instances, adding or removing ontology classes, or completely remove them. Also, when she selects an annotation from the list (regarding a part of a document), the document scrolls to the corresponding part, which is highlighted with the same color as the annotation instance.

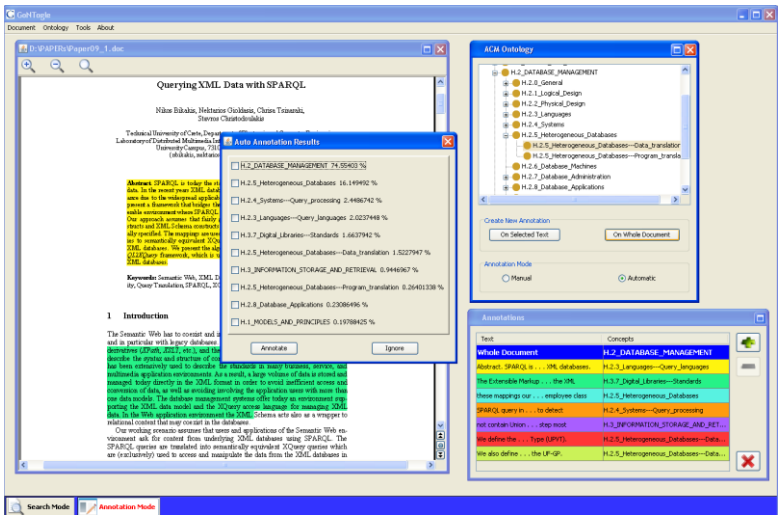

Figure 4. Semantic annotation example

4.3 Implementation

In what follows we provide technical information about the implementation of our system. All annotation and search facilities presented in this paper have been implemented in a Java prototype. Application screenshots, as well as the application

itself and installation instructions can be found in *http://web.imis.athena-innovation.gr/~dalamag/gontogle*. A demonstration of GoNTogle tools is presented in [23].

To develop our system, we used several open source tools and libraries. For indexing and keyword searching we used the Lucene search engine library. Lucene modules participate in several components of our system: a) Document text indexing for search purposes (*Indexing Component*). b) Document retrieval and scoring regarding textual similarity (*Search Component*). c) Indexing and querying documents for automatic annotation purposes (*Semantic Annotation Component*).

We used the Protégé[3] server and MySQL database for the *Ontology Server Component*, so that document annotations are stored as class instances. Through Protégé API, for each annotation, we store information that is required for processes such as retrieval of the specific annotation, ontology search scoring for a specific class-document pair, etc.

OpenOffice API[4] was essential in incorporating in our system a viewer that could maintain the exact format of *.doc* documents, which is a very common filetype. The same applies for Multivalent[5], a generalized document viewer that was integrated in our system so that *.pdf* files could also maintain their format when being viewed and annotated.

5 Evaluation

In this section, we present the experiments we performed in order to evaluate the effectiveness of our methods. In Section 5.1 we present the evaluation of the automatic annotation method. In Section 5.2, we compare our proposed hybrid search method with keyword-based and semantic-based search .

5.1 Automatic Annotation

In order to demonstrate the effectiveness of the proposed automatic annotation method, we perform a user-based evaluation. The effectiveness of our method is validated in terms of *Precision at position n* (*P@n*) and *Recall*.

Configuration

We turned the ACM Computing Classification into an OWL ontology. The ontology produced is a 4-level structure with 1463 nodes. First, we performed an initial set of experiments in order to compare the simple *kNN* and the *weighted kNN* classification methods and also to indentify the best value for the *k* factor. Best precision and recall values were observed for *k=7* using the *weighted kNN* algorithm.

[3] http://protege.stanford.edu/

[4] http://api.openoffice.org/

[5] http://multivalent.sourceforge.net/

Moreover, the weights used for the automatic annotation method (Section 2.1), w_1 and w_2 are calculated at *0.6* and *0.4* respectively after tuning. Intuitively, these values suggest that, in our problem setting, textual similarity is slightly more important than semantic similarity in case of automatic annotation.

Evaluation Scenario

We asked from 15 users (PhD students and researchers in various areas of computer science) to participate in our experimental evaluation. Each user selected 2 areas of her research interests and for each area she collected 10 research papers that she was familiar with. In order to train our system, we asked from each user to annotate (parts or/and the whole of) 12 out of her 20 papers with at least one ACM class, using the GoNTogle framework.

After every user had performed the training task, we asked each of them to evaluate the automatic annotation suggestions provided by GoNTogle, for the remaining 8 papers of each user (test set). Note that, before reviewing the system suggestions, each user was asked which annotation classes she expected to be given by the system. The system presented a ranked list of annotation classes and each user was required to check the valid ones. Also, each user should point out valid classes that were not found between the system suggestions, as well as valid classes that, even they had not thought of, the system correctly suggested them.

Based on the data collected, we calculated the *Precision at position n* (*P@n*) and *Recall* values for each user separately, as well as the mean average values for all users. Also, for correctly suggested annotation classes that the user had not initially thought of using them, we introduce the measure of *Unexpected Valid Class Suggestion (UVCS),* defined as follows:

$$UVCS = \#Correctly\ suggested\ and\ not\ initially\ though\ classes$$

P@n and *Recall* are defined as follows: $\text{P@n} = \frac{\#\text{relevant results in top n suggestions}}{\text{n}}$ and $Recall = \frac{\#relevant\ results\ in\ suggestions}{\#relevant\ results}$, where we count as *relevant results*, the ACM classes considered valid by the user.

Evaluation Results

Table 1 presents, for each user, the average *P@n* values, for her 8 automatically annotated papers. In addition, the average *Recall* (regarding the top-5 results) and the average *UVCS* values are presented at Table 2.

Note that, due to our annotation scenario (annotating research papers with ACM classes), it is rational to regard only the top-5 results during the *P@n* computation. That is, because the majority of the research papers under consideration do not handle more than 5 ACM hierarchy topics.

As we can observe, our method achieves high values both for *Precision* and *Recall* metrics. Moreover high *Recall* values have been achieved, with an average *Recall* value equal to 0.93. We should note that the relatively low *P@4 and P@5* are justified from the fact that, for a respectable amount of test documents, the users expected (and thus validated) no more than 1-3 classes, that were found in the top 3 positions of the system's ranked suggestion list. Finally, it is obvious from the *UVCS* metric, that the automatic annotation mechanism supports and guides users during the

annotation process, by suggesting correct classes that users had not previously thought of.

Table 1. The average *Precision at position n* (*P@n*) for each user

User	P@1	P@2	P@3	P@4	P@5
1	0.82	0.79	0.79	0.75	0.68
2	1.00	0.94	0.80	0.65	0.60
3	0.80	0.80	0.70	0.70	0.76
4	1.00	1.00	0.80	0.84	0.80
5	1.00	0.90	0.90	0.82	0.81
6	0.80	0.90	0.73	0.70	0.64
7	1.00	1.00	0.93	0.85	0.84
8	0.93	1.00	0.73	0.71	0.69
9	0.90	0.90	0.87	0.80	0.76
10	0.91	0.87	0.80	0.75	0.71
11	1.00	1.00	0.87	0.84	0.78
12	0.80	0.77	0.72	0.70	0.66
13	0.95	0.92	0.83	0.75	0.68
14	1.00	0.90	0.87	0.80	0.76
15	0.80	0.80	0.73	0.65	0.56
Avg	0.91	0.90	0.81	0.75	0.72

Table 2. The average *Recall* and the average *UVCS* for each user

User	Recall	UVCS
1	0.80	0.40
2	0.92	0.20
3	0.98	0.20
4	0.97	0.40
5	0.98	0.40
6	1.00	1.20
7	0.97	0.20
8	0.82	0.20
9	1.00	0.20
10	0.89	1.00
11	0.88	0.80
12	0.95	0.65
13	0.87	0.40
14	0.95	1.60
15	1.00	0.00
Avg	0.93	0.52

5.2 Search

In this section, we present an evaluation comparing the effectiveness of the search types provided by our framework. The comparison is performed in terms of *Precision at position n*, *Recall, F-measure* and *Precision-Recall curve*. In all cases, the proposed hybrid search method delivers higher quality results than traditional keyword-based or semantic-based search methods.

Configuration

The weights used for the hybrid search method (Section 3.1) are assigned the following values: $w_3=0.7$, $w_4=0.3$ and $w_5=0.6$, $w_6=0.4$ after tuning. Intuitively, these values suggest that, in our problem setting, semantic-based score is slightly more important than keyword-based score in hybrid search.

Evaluation Scenario

Our corpus consists of the 300 manually and automatically annotated research papers from the previous experiment (Section 5.1). First, we collect all the keywords defined in these papers and we randomly choose 10 keywords to be used as queries. Note that, keywords queries may contain one or more tokens.

Also, we map the selected keyword queries to semantic queries, using the ontology classes. That is, to construct semantic queries that correspond to the keyword ones, we select the ontology classes that are most similar to the keyword content. In this way, we are able to perform both keyword, and ontology search, as well as hybrid search, comparing the effectiveness of each approach.

Table 3 presents the 10 keyword queries (q_{key}) which are used for this experiment. Table 4, presents the corresponding semantic queries (q_{sem}) expressed using the classes from ACM ontology. Hybrid queries are expressed by the combination of a keyword query and its corresponded semantic query. For hybrid search we apply booth (*OR, AND*) Boolean operators. The hybrid queries applying *AND* and *OR* operators are denoted respectively as q_{hybrA} and q_{hybrO}.

Table 3. Keyword Queries

ID	Keywords
$q_{key}1$	knowledge discovery and privacy
$q_{key}2$	stream mining
$q_{key}3$	RDF indexing
$q_{key}4$	spatial databases
$q_{key}5$	clustering
$q_{key}6$	spatial access
$q_{key}7$	query language
$q_{key}8$	data model
$q_{key}9$	XML interoperability
$q_{key}10$	information integration

Table 4. Semantic Queries

ID	Classes
$q_{sem}1$	K.4.1 [Public Policy Issues]: Privacy
$q_{sem}2$	H.2.8 [Database Applications]: Data mining
$q_{sem}3$	H.3.1 [Content Analysis and Indexing]: Indexing methods
$q_{sem}4$	H.2.8 [Database Applications]: Spatial databases and GIS
$q_{sem}5$	H.3.3 [Information Search and Retrieval]: Clustering
$q_{sem}6$	H.2.2 [Physical Design]: Access Methods
$q_{sem}7$	H.2.3 [Languages]: Query languages
$q_{sem}8$	H.2.1 [Logical Design]: Data models
$q_{sem}9$	D.2.12 [Interoperability]
$q_{sem}10$	H.2.5 [Heterogeneous Databases]

For each query we measure the quality of retrieval method using the *Precision at position n at position n,* for *n* [1 to10] and *Recall*. Based on these measures, we compare the various search types offered by our system: a) *Keyword-based* search, b) *Semantic-based* search, c) *Hybrid* search using *AND* operator (*hybrA*) and d) *Hybrid* search using *OR* operator (*hybrO*). Finally, for each search type, we compute the average *Precision* at positions 1 to 10, *Recall, F-measure* and *Precision-Recall curves* for all queries.

Average Evaluation Results For All Queries

Table 5 presents the average *P@n* for *n* [1 to 10] and the average *Recall* and *F-measure* values for all queries. Note that, most queries in hybrid search using the *AND* operator, do not retrieve more than 5-6 documents (as we can see from Table 6). As a consequence, the precision, for this search type is calculated only at positions 1 to 5.

Precision. As we can observe from Table 5, the hybrid search (for both operators) outperforms the keyword-based and semantic-based search at every position, with *hybrA* achieving slightly higher values at positions 4 and 5. Moreover, we can see that keyword-based search radically decreases after position 4, where semantic-based and hybrid search start decreasing progressively after the 6th position.

Hybrid search compared to keyword-based search, achieves a maximum increase of 100% at position 7 and a minimum increase of 33.3% at position 2. Comparing hybrid with semantic-based search, hybrid, achieves a maximum increase of 17.2% at position 10 and a minimum increase of 0% at position 1.

Recall. As we can see, the *hybrO* outperforms the keyword-based and semantic-based search, achieving recall value close to 1 (0.98). Moreover, *hybrA* achieves slightly lower recall values than semantic-based search. This is due to the fact that

hybrA search is very restrictive. So, too few documents are returned for each query with negative influence on the recall values.

Comparing *hybrO* with keyword-based search, *hybrO*, achieves an increase of 78.2%. Moreover, despite the low recall values of *hybrA* method, in comparison with keyword-based search, it increases the recall value at 20%. In comparison with semantic-based search, *hybrO* achieves a increase of 16.7%. Finally, *hybrA* achieves lower recall values than semantic-based search, having a decrease of 21.4%.

F-measure. As we can see, the hybrid search outperforms the other methods in F-measure value. Comparing *hybrO* with keyword-based and semantic-based search, *hybrO* achieves an increase of 77% and 16.4% respectively. Moreover, comparing *hybrA* with keyword-based and semantic-based search, *hybrA* achieves an increase of 52% and 0% respectively.

Table 5. The average *Precision at position n* (*P@n*), *Recall* and *F-measure* for all queries

	P@1	P@2	P@3	P@4	P@5	P@6	P@7	P@8	P@9	P@10	Recall	F-measure
q_{key}	0.70	0.75	0.70	0.70	0.60	0.52	0.47	0.48	0.44	0.43	0.55	0.48
q_{sem}	1.00	0.95	0.90	0.88	0.90	0.87	0.81	0.76	0.70	0.64	0.84	0.73
q_{hybrA}	1.00	1.00	1.00	1.00	0.98	-	-	-	-	-	0.66	0.73
q_{hybrO}	1.00	1.00	0.97	0.98	0.96	0.95	0.94	0.89	0.81	0.75	0.98	0.85

Precision vs. Recall. Figure 5 shows the average precision-recall curve for all queries. As we can see, hybrid search has a very stable performance, achieving high precision (close to 1) even for recall values greater than 0.8. *hybrO* precision starts to decrease noticeably only after recall values are greater than 0.9. For recall values lower than 0.6, *hybrA* achieves precision values higher than *hybrO*. Semantic-based search precision, progressively decreases from the beginning while recall increases. Finally, keyword-based search precision values rapidly decrease for recall values greater than 0.4.

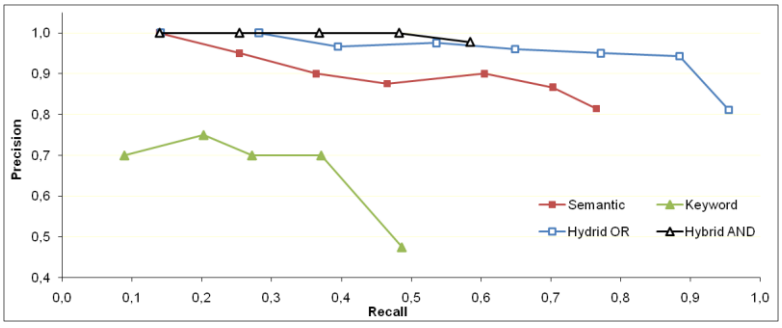

Figure 5. The average precision-recall curve for all queries

Evaluation Results For Each Query

Figure 6 presents for each query, the *P@n* for *n* [1 to 10] and *Recall* values.

As we can see, in all queries, the hybrid search (for both boolean operators) outperforms the keyword-based and semantic-based search in precision values at every position. Moreover, regarding the recall measures, the *hybrO* search outperforms the other search methods in every query, with 9 out of 10 queries achieving recall values equal to 1.

As far as *P@n* is concerned, hybrid search achieves the highest precision values for all queries in every position. Hybrid search using *AND* and *OR* operators achieve similar precision values. However in many cases *AND* operator returns less than 10 documents. Semantic-based search achieves lower precision values (except *hybrA* for Query 6) than hybrid search, and higher values than keyword-based search (with 3 exceptions, Queries 4,5,6). Finally, keyword-based search achieves, in general, the lowest precision values.

As far as *recall* is concerned, hybrid search using *OR* operator achieves the highest recall values in all queries, with 9 out of 10 queries achieving recall values equal to 1. Semantic-based search achieves lower recall values than the former and higher or equal than rest methods, with two exceptions (Queries 6,8). Moreover, hybrid search using *AND* operator achieves lower or equal recall values than semantic-based search and higher than keyword-based search (with one exception, Query 2). Finally, keyword-based search achieves, in general, lowest recall values.

	Query 1				**Query 2**				**Query 3**				**Query 4**			
	$q_{key}1$	$q_{sem}1$	$q_{hybrA}1$	$q_{hybrO}1$	$q_{key}2$	$q_{sem}2$	$q_{hybrA}2$	$q_{hybrO}2$	$q_{key}3$	$q_{sem}3$	$q_{hybrA}3$	$q_{hybrO}3$	$q_{key}4$	$q_{sem}4$	$q_{hybrA}4$	$q_{hybrO}4$
P@1	1.00	1.00	1.00	1.00	0	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
P@2	1.00	1.00	1.00	1.00	0.50	0.50	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
P@3	1.00	1.00	1.00	1.00	0.33	0.67	1.00	0.67	0.67	0.67	1.00	1.00	1.00	0.67	1.00	1.00
P@4	1.00	1.00	1.00	1.00	0.50	0.50	-	0.75	0.75	0.75	1.00	1.00	1.00	0.75	1.00	1.00
P@5	1.00	1.00	1.00	1.00	0.40	0.60	-	0.60	0.60	0.80	1.00	1.00	1.00	0.80	1.00	1.00
P@6	0.83	1.00	-	1.00	0.33	0.67	-	0.67	0.50	0.67	-	0.83	0.83	0.83	1.00	1.00
P@7	0.71	1.00	-	1.00	0.29	0.57	-	0.57	0.43	0.71	-	0.86	0.71	0.71	-	1.00
P@8	0.63	1.00	-	1.00	0.25	0.50	-	0.50	0.38	0.63	-	0.75	0.75	0.75	-	1.00
P@9	0.56	1.00	-	1.00	0.22	0.44	-	0.44	0.33	0.56	-	0.67	0.67	0.78	-	0.89
P@10	0.50	0.90	-	1.00	0.20	0.40	-	0.40	0.40	0.50	-	0.60	0.60	0.80	-	0.80
Recall	0.45	0.82	0.45	0.91	0.50	1.00	0.25	1.00	0.67	0.83	0.83	1.00	0.75	1.00	0.75	1.00

	Query 5				**Query 6**				**Query 7**				**Query 8**			
	$q_{key}5$	$q_{sem}5$	$q_{hybrA}5$	$q_{hybrO}5$	$q_{key}6$	$q_{sem}6$	$q_{hybrA}6$	$q_{hybrO}6$	$q_{key}7$	$q_{sem}7$	$q_{hybrA}7$	$q_{hybrO}7$	$q_{key}8$	$q_{sem}8$	$q_{hybrA}8$	$q_{hybrO}8$
P@1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0	1.00	1.00	1.00	0	1.00	1.00	1.00
P@2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0	1.00	1.00	1.00	0	1.00	1.00	1.00
P@3	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.33	1.00	1.00	1.00	0	1.00	1.00	1.00
P@4	1.00	0.75	1.00	1.00	1.00	1.00	1.00	1.00	0.25	1.00	1.00	1.00	0	1.00	1.00	1.00
P@5	0.80	0.80	1.00	1.00	0.80	1.00	0.80	1.00	0.20	1.00	1.00	1.00	0	1.00	1.00	1.00
P@6	0.67	0.83	-	1.00	0.67	0.83	0.67	1.00	0.33	1.00	1.00	1.00	0	1.00	1.00	1.00
P@7	0.57	0.71	-	1.00	0.71	0.71	0.57	1.00	0.29	1.00	1.00	1.00	0.14	1.00	-	1.00
P@8	0.63	0.63	-	0.88	0.75	0.63	0.50	1.00	0.38	1.00	1.00	1.00	0.13	0.88	-	0.88
P@9	0.56	0.56	-	0.78	0.67	0.56	0.44	0.89	0.44	0.89	1.00	1.00	0.11	0.78	-	0.78
P@10	0.50	0.50	-	0.70	0.60	0.50	0.40	0.80	0.50	0.80	0.90	1.00	0.10	0.70	-	0.70
Recall	0.63	0.63	0.63	0.88	0.67	0.63	0.50	1.00	0.50	0.80	0.90	1.00	0.14	0.70	0.89	1.00

	Query 9				**Query 10**			
	$q_{key}9$	$q_{sem}9$	$q_{hybrA}9$	$q_{hybrO}9$	$q_{key}10$	$q_{sem}10$	$q_{hybrA}10$	$q_{hybrO}10$
P@1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
P@2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
P@3	1.00	1.00	1.00	1.00	0.67	1.00	1.00	1.00
P@4	0.75	1.00	1.00	1.00	0.75	1.00	1.00	1.00
P@5	0.60	1.00	1.00	1.00	0.60	1.00	1.00	1.00
P@6	0.50	1.00	-	1.00	0.50	0.83	1.00	1.00
P@7	0.43	0.86	-	1.00	0.43	0.86	-	1.00
P@8	0.38	0.75	-	0.88	0.50	0.88	-	1.00
P@9	0.33	0.67	-	0.78	0.56	0.78	-	0.89
P@10	0.30	0.60	-	0.70	0.60	0.70	-	0.80
Recall	0.43	0.86	0.71	1.00	0.75	0.88	0.75	1.00

Figure 6. The *Precision at position n* (*P@n*) and the *Recall* for each query

6 Related Work

A great number of approaches on semantic annotation have been proposed in the literature [6, 7]. Most of them are focused on annotating web resources such as HTML pages [8, 9, 10, 11, 12, 13, 14].

As far as plain text (or HTML) annotation is concerned, there are approaches that differ in the annotation and search facilities they offer. GATE [15] is a platform that offers an architecture, a framework and a graphical tool for language processing. Tools and resources are offered to perform textual annotation both manually and automatically using information extraction (IE) techniques.

KIM [16] provides an infrastructure for semantic annotation of documents (text or HTML), restricted, however, to its own ontology, called KIMO. The information extraction, document management and annotation part is based on GATE. The aim of the IE engine is the recognition of named entities with respect to the KIMO ontology. Compared to the above approaches, GoNTogle provides advanced searching facilities using a flexible combination of keyword-based and semantic-based search over documents. Also, it provides automatic annotation facilities based on models trained from user annotation history, so that annotation suggestions are tailored to user behavior.

AKTiveMedia [17] supports the annotation of text, images and HTML documents using both ontology-based and free-text annotations. For the automatic annotation task an underlying information extraction (IE) system has been integrated, learning from previous annotations and suggests annotations to the user. However, AKTiveMedia does not provide search facilities. Furthermore, the supported automatic annotation mechanism provides very low performance, when annotations concerns more than one tokens (due to the IE system). In addition, a serious limitation of the automatic annotation mechanism is that it takes into consideration only one class per annotation. In case of annotations with multiples classes, the rest of the classes are skipped.

The above tools support annotations on HTML or plain text. As far as popular document formats are concerned, PDFTab [18] is a Protégé plug-in for annotating PDF documents with OWL ontologies classes. Annotations are stored in the internal document representation, with the document structure remaining unchanged. Compared to GoNTogle, PDFTab has several limitations: it does not provide any search facilities or automatic annotation method. SemanticWord [19] is a MS Word plug-in which offers MS Word annotations with DAML+OIL ontologies. Compared to GoNTogle, SemanticWord integrates an information extraction system with no learning support to suggest annotations. Also, SemanticWord does not provide search facilities and does not support OWL and RDF/S ontologies.

Regarding the semantic search, in the recent years, numerous systems and approaches have been proposed in the literature [20]. An approach close to our, is introduced at [21], where a combination of keyword and semantic search over web sources is supported, on top of the AKTiveMedia framework [17]. A noticeable drawback of this approach is that the ranking of hybrid search, is relying only at keyword search where the semantic part is utilized only to exclude or include a result and not to rank it. Moreover, [21] does not support advanced search operations related to ontology semantics. Additionally, an interesting but less relative approach [22],

analyzes the meaning of words and phrases, to define semantic relations between lexicalized concepts. In that case, syntactic search is extended with semantics, by converting words into concepts and exploiting the arisen semantics.

7 Conclusion and Future Work

In this paper we presented GoNTogle, a framework for document annotation and retrieval, built on top of Semantic Web and IR technologies. GoNTogle supports both manual and automatic document annotation using ontologies. A learning mechanism is implemented, providing automatic document annotation facilities based on textual information and user annotation history.

In order to overcome the drawbacks of traditional keyword-based (like concept polysemy and synonymy) and semantic-based search (like partial or not existing annotations) we propose a hybrid search method. Hybrid search provides a flexible combination of keyword-based and semantic-based search. Moreover, several advanced ontology-based search operations are provided. Ontology information is exploited, to help the user expand or shrink the resulting list in order retrieve high quality results.

A user-based evaluation is performed, in order to demonstrate the effectiveness of the automatic annotation method. Moreover, a comparative evaluation validates that, the proposed hybrid search, outperforms in all cases the keyword-based and semantic-based search in terms of precision and recall.

Finally, all the proposed methods are implemented as a fully functional tool.

Our future work involves: a) Supporting more knowledge representation forms (e.g. Mind maps). b) Adding advanced search facilities exploiting ontology reasoning techniques. c) Integrating several semantic-based natural language techniques. d) Studying how tagging techniques can be integrated to GoNTogle framework. e) Using GoNTogle framework to support the clipping department of an organization or a company in order to perform extended experiments in large corpora. f) Adapting the framework to commercial document viewers (MS Word and Adobe Reader).

Acknowledgments. We would like to thank the all the PhD students and the research staff from *IMIS R.C. "Athena"* and *KDBS Lab* (*NTUA*) for their contribution in the evaluation part of this work. Finally, we would also like to thank Dimitris Sacharidis from *IMIS/R.C. "Athena"* for many helpful comments on earlier versions of this article.

References

1. Mitchell T.M. "Machine Learning" WCB/McGraw-Hill, 1997.
2. Handschuh, S., Staab, S. (eds.): "Annotation for the Semantic Web". IOS Press, (2003)

3. Agosti, M., Ferro, N.: "A Formal Model of Annotations of Digital Content". ACM Transactions on Information Systems (TOIS) 26(1), 3:1–3:57 (2008)
4. Agosti M., Albrechtsen H., Ferro N., Frommholz I., Hansen P., (et.al): "DiLAS: a digital library annotation service". In Proc. of IWAC 2005.
5. Haslhofer B., Jochum W., King R., Sadilek C., Schellner K.: "The LEMO annotation framework: weaving multimedia annotations with the web". JODL 10(1):15-32 (2009)
6. Reeve L., Han H.: "Survey of semantic annotation platforms". In Proc. of the ACM Symposium on Applied Computing '05.
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 Semantics, vol. 4, 2006.
8. Kiyavitskaya N., Zeni N., Cordy J.R., Mich L., Mylopoulos J.: "Cerno: Light-weight tool support for semantic annotation of textual documents". Data Knowl. Eng. (DKE) 68(12) (2009)
9. Hogue A., Karger D.: "Thresher: automating the unwrapping of semantic content from the World Wide Web". In Proc. of WWW 2005.
10. Cimiano P., Handschuh S., Staab S.: "Towards the self-annotating web". In Proc. of WWW 2004.
11. 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) (2003).
12. SMORE: Create OWL Markup for HTML Web Pages. *http://www.mindswap.org/2005/SMORE/.*
13. Handschuh, S., Staab, S., Ciravegna, F.: "S-CREAM: Semi-automatic CREAtion of Metadata". In Proc. of EKAW 2002.
14. Vargas-Vera, M., Motta, E., Domingue, J, Lanzoni (et.al) : "MnM: Ontology Driven Semi-automatic and Automatic Support for Semantic Markup" In Proc. of EKAW 2002.
15. Cunningham H., Maynard D., Bontcheva K., Tablan V.: "GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications". In Proc. of the ACL 2002.
16. Kiryakov A., Popov B., Terziev I., Manov D., Ognyanoff D.: "Semantic annotation, indexing, and retrieval". Journal of Web Semantics 2(1), 2004.
17. Chakravarthy A., Lanfranchi V., Ciravegna F.:"Cross-media document annotation and enrichment". In 1st Semantic Authoring and Annotation Workshop 2006.
18. Eriksson H.: "An annotation tool for semantic documents". In Proc. of the ESWC 2007
19. Tallis M., "SemanticWord processing for content authors": In Proc. of the Knowledge Markup and Semantic Annotation Workshop 2003.
20. Mangold C., "A survey and classification of semantic search approaches", Int. J. Metadata Semantics and Ontology 2 (1) (2007).
21. Bhagdev, R., Chapman, S., Ciravegna, F., Lanfranchi, V., Petrelli, D.: "Hybrid search: Effectively combining keywords and semantic searches". In Proc. of ESWC 2008.
22. Giunchiglia F., Kharkevich U., Zaihrayeu I., "Concept search", In Proc. of ESWC 2009
23. Giannopoulos G., Bikakis N., Dalamagas T., Sellis T.: "GoNTogle: A Tool for Semantic Annotation and Search". In Proc. of the ESWC 2010 (Demo).