GoNTogle: A Tool for Semantic Annotation and Search

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

[1] KDBSL Lab | School of ECE | Nat. Tech. Univ. of Athens | Greece
giann@dblab.ntua.gr, bikakis@dblab.ntua.gr

[2] Institute for the Management of Information Systems | "Athena" Research Center | Greece
dalamag@imis.athena-innovation.gr, timos@imis.athena-innovation.gr

Abstract. This paper presents GoNTogle, a tool which provides advanced document annotation and search facilities. GoNTogle allows users to annotate several document formats, using ontology concepts. It also produces automatic annotation suggestions based on textual similarity and previous document annotations. Finally, GoNTogle combines keyword and semantic-based search, offering advanced ontology query facilities.

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

1 Introduction

Semantic annotation and search tools are at the core of Semantic Web Technology. Annotations involve tagging of data with concepts (i.e., ontology classes) so that data becomes meaningful. Annotating data can help in providing better search facilities, since it helps users to search for information not only based on the traditional keyword-based search, but also using well-defined general concepts that describe the domain of their information need.

A great number of approaches on semantic annotation have been proposed in the literature [2]. Most of them are focused on annotating web resources such as html pages or plain text [3,4,5,6]. As far as popular document formats are concerned, there are approaches that differ in the annotation and search facilities they offer [7,8].

In this paper we present GoNTogle. GoNTogle supports manual and automatic annotation of several types of documents (doc, pdf, rtf, txt, odt, sxw) using ontology classes, in a fully collaborative environment. It also provides searching facilities beyond the traditional keyword-based search, using a flexible combination of keyword and semantic-based search. In contrast with other works, our aim was to implement an easy-to-use document annotation and search tool, that would fully support (a) viewing and annotating popular document types while maintaining their initial format, (b) sharing those annotations and (c) searching for documents combining keyword and semantic-based search.

The key features of our tool are the following:

- It allows users to open and view widely used document formats such as *.doc* and *.pdf* , maintaining their original format.

- It provides an easy and intuitive way of annotating documents (or document parts) using OWL and RDF/S ontologies.
- It provides an automatic annotation mechanism based on models trained from user annotation history, so that annotation suggestions are tailored to user behavior.
- It is based on a server-based architecture, where document annotations are stored in a central repository. Thus, we offer a collaborative environment where users can annotate and search documents.
- It combines keyword and semantic search, providing advanced search facilities for both types of search.

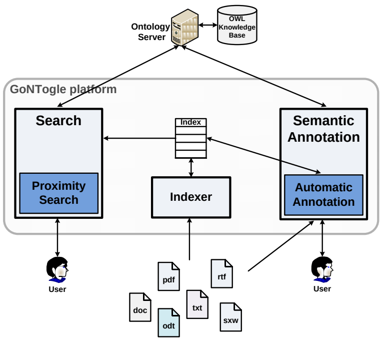


Figure 1. GoNTogle architecture

2 System Overview

GoNTogle's architecture is presented in Figure 1. The system is divided into 4 basic components: a) *Semantic Annotation Component*, that provides facilities regarding the semantic annotation of documents. It consists of 3 modules: a Document Viewer, an Ontology Viewer and an Annotation Editor. b) *Ontology Server Component*, that stores the semantic annotations of documents in the form of OWL ontology instances. c) *Indexing Component*, that is responsible for indexing the documents using an inverted index. d) *Search Component*, that allows users to search for documents using both textual (keyword search) and semantic (ontology search) information.

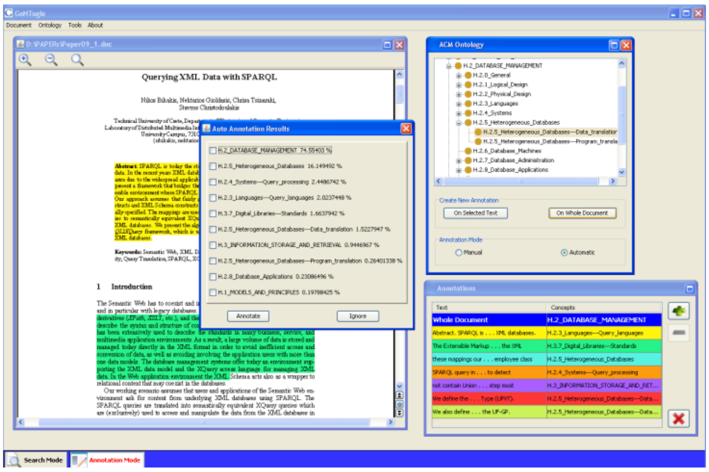
2.1 Semantic Annotation

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

Figure 2 depicts the Semantic Annotation window of our application. The user may open a document in the Document Viewer, maintaining its original format. In the particular example, the user has opened a *.doc* document. Moreover, she can load and view the hierarchy of an ontology through the Ontology Viewer.

The user can, then, select one or more ontology classes and manually annotate the whole document or part of it. The annotation is saved as an ontology instance in the Ontology Server, along with information about the annotated document (or part). On 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 2), 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 edit those annotation instances, adding or removing ontology classes, or completely remove them.

The user may also choose the automatic annotation functionality. In this case, the system suggests candidate ontology classes, executing our learning method based on *weighted kNN* classification [1], that exploits user annotation history to automatically suggest annotations.



Auto Annotation Results

- □ H.2_DATABASE_MANAGEMENT 74.55403 %
- □ H.2.5_Heterogeneous_Databases 16.149492 %
- □ H.2.4_Systems---Query_processing 2.4486742 %
- □ H.2.3_Languages---Query_languages 2.0237448 %
- □ H.3.7_Digital_Libraries---Standards 1.6637942 %
- □ H.2.5_Heterogeneous_Databases---Data_translation 1.5227947 %
- □ H.3_INFORMATION_STORAGE_AND_RETRIEVAL 0.9446967 %
- □ H.2.5_Heterogeneous_Databases---Program_translation 0.26401338 %
- □ H.2.8_Database_Applications 0.23086496 %
- □ H.1_MODELS_AND_PRINCIPLES 0.19788425 %

Annotate Ignore

Annotation Mode: ○ Manual ● Automatic

Text	Concepts
Whole Document	H.2_DATABASE_MANAGEMENT
Abstract. SPARQL is . . . XML databases.	H.2.3_Languages---Query_languages
The Extensible Markup . . . the XML	H.3.7_Digital_Libraries---Standards
these mappings our . . . employee class	H.2.5_Heterogeneous_Databases
SPARQL query in . . . to detect	H.2.4_Systems---Query_processing
not contain Union . . . step most	H.3_INFORMATION_STORAGE_AND_RET...
We define the . . . Type (UPVT).	H.2.5_Heterogeneous_Databases---Data...
We also define . . . the UP-GP.	H.2.5_Heterogeneous_Databases---Data...

Figure 2. Semantic annotation example

2.2 Search

Search component provides a series of search facilities which are described below. We start with the simple ones:

- *Keyword-based search*. This is the traditional searching model. The user provides keywords and the system retrieves relevant documents based on textual similarity.
- *Semantic-based search*. The user navigates through the classes of the ontology and selects one or more of them. The result list from this type of search consists of all documents that have (partially or on their whole) been annotated with one or more of the chosen classes.
- *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.

Next we present a set of advanced searching facilities 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 which have been annotated with a class that also annotates *d*.
- *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 which are already in the result list and have been annotated with a class that also annotates *d*.
- *Get Next Generation*. The resulting list from an semantic-based search can be expanded by propagating the search on lower levels in the ontology (i.e., if class *c* has been used, then search is propagated only in direct subclasses of *c*). This is the case when the search topic is too general.
- *Get Previous Generation*. This offers the inverse functionality of the previous option. The resulting list from an semantic-based search can be expanded by propagating the search on higher levels in the ontology (i.e., if class *c* has been used, then search is propagated only in direct superclasses of *c*). This is the case when a search topic is too narrow.
- *Proximity Search*. This search option allows the user to search for documents not only belonging to a selected class but also belonging to the classes's subclasses. That is, if class *c* has been used, then search is propagated in all direct subclasses of *c* and their direct subclasses. The resulting documents gathered from those three levels of the ontology hierarchy are weighted properly (documents from the initial class *c* get higher score).

The resulting documents are presented in a ranked list. From the information presented in this list the user can find out whether a resulting document comes from keyword/ontology/hybrid search, and which classes it is annotated with (if any). Also, each result is accompanied by a score. This score is a weighted sum of (a) the textual similarity score given by keyword search and (b) a score that represents the extend to which each document is annotated with the classes selected in ontology search.

3 Implementation and Evaluation

In what follows we provide technical information about the implementation of our system and we present a preliminary experiment we performed.

Implementation. To compose our system, we utilized several open source tools and libraries. For indexing and keyword searching we used Lucene search engine library[1].

We used Protégé[2] as Ontology Server, so that document annotations are stored as OWL class instances.

OpenOffice API[3] 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[4], a generalized document viewer that was integrated in

[1] http://lucene.apache.org/java/docs/

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

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

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

our system so that *.pdf* files could also maintain their format when being viewed and annotated.

All annotation and search facilities presented in this paper have been implemented in a Java prototype. Detailed information and application screenshots, as well as the application itself and installation instructions can be found in http://web.imis.athena-innovation.gr/~dalamag/gontogle.

Evaluation. We have performed a preliminary evaluation of the semantic annotation tool. More specifically, we tested the precision and recall of the automatic annotation process.

We turned the ACM Computing Classification into an OWL ontology. The ontology produced is a 4-level structure with 1463 nodes.

We used 500 papers, pre-categorized according to the ACM System, as sample documents for the ontology classes. Then, we used the tool to automatically annotate 66 papers from the publication database maintained in our lab. Figure 4 presents precision and recall diagrams for simple and weighted *kNN*. Best values were observed for *k=7*.

We observed better results when using weighted *kNN* compaired to simple *kNN*, so we adopted the former in our final system implementation.

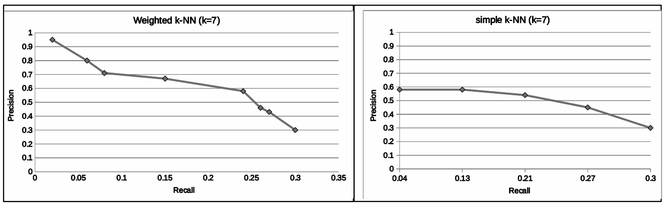


Figure 3. Evaluation results for automatic annotation based on *kNN*

References

1. Mitchell T.M. "Machine Learning" WCB/McGraw-Hill, 1997.
2. 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.
3. 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 '02.
4. 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'02.
5. Kiryakov A., Popov B., Terziev I., Manov D., Ognyanoff D.: "Semantic annotation, indexing, and retrieval". Journal of Web Semantics 2(1), 2004.
6. Chakravarthy A., Lanfranchi V., Ciravegna F.:"Cross-media document annotation and enrichment". In 1st Semantic Authoring and Annotation Workshop '06.
7. Eriksson H.: "An annotation tool for semantic documents". In Proc. of the ESWC '07
8. Tallis M., "SemanticWord processing for content authors": In Proc. of the Knowledge Markup and Semantic Annotation Workshop '03.