A
Dissertation
on

Design And Implementation of Semantic Search Tool With Automatic Document Classification and Summarization

Submitted in partial fulfilment of the requirement
For the award of Degree of

MASTER OF ENGINEERING
(Computer Technology and Application)

Submitted by

NITIN NIZHAWAN
(University Roll No. 10191)

Under the Guidance of
Dr. DAYA GUPTA

DEPARTMENT OF COMPUTER ENGINEERING
DELHI COLLEGE OF ENGINEERING
BAWANA ROAD, NEW DELHI
(DELHI UNIVERSITY)
June-2008

ABOUT ORGNIZATION

About NIC

National Informatics Center (NIC)

National Informatics Centre (NIC) of the Department of Information Technology is providing network backbone and e-Governance support to Central Government, State Governments, UT Administrations, Districts and other Government bodies. It offers a wide range of ICT services including Nationwide Communication Network for decentralized planning, improvement in Government services and wider transparency of national and local Governments. NIC assists in implementing Information Technology Projects, in close collaboration with Central and State Governments, in the areas of (a) Centrally sponsored schemes and Central sector schemes, (b) State sector and State sponsored projects, and (c) District Administration sponsored projects. NIC endeavours to ensure that the latest technology in all areas of IT is available to its users.

The organizational set up of NIC encompasses its Headquarters at New Delhi, State Units in all the 28 State capitals and 7 Union Territory Headquarters and District centers in almost all the Districts of India. The Organization employs a large pool of efficient technical manpower.

At the NIC Headquarters, a large number of Application Divisions exist which provide total Informatics Support to the Ministries and Departments of the Central Government. NIC computer cells are located in almost all the Ministry Bhawans of the Central Government and Apex Offices including the Prime Minister's Office, the Rashtrapati Bhawan and the Parliament House.

About DCE

Delhi College of Engineering

With a history stretching back over 65 years, providing technical education within a modern educational environment and strong academic staff, DCE is strongly identified with engineering education in India. Since its inception and foundation, DCE has constantly lead the way in reform movements, and in the latter era of the Republic of India, DCE has assumed pivotal roles in the reconstruction, modernization, and administration of the society. The efforts and expertise of DCE graduates have been major contributors in the planning and construction of India's infrastructure. DCE is an institution which defines and continues to update methods of engineering and architecture in India. It provides its students with modern educational facilities while retaining traditional values, as well as using its strong industrial contacts to mold young, talented individuals who can compete in the global arena. The aim of DCE is to rank among leading universities globally. Consequently, DCE's mission is to educate individuals to be competitive not only in India, but all over the world. Within an intensely competitive environment, the college has adopted a dynamic, global, high-quality, creative and communicative approach in education, as well as research and development. Keeping abreast with modern developments, DCE is constantly restructuring itself and renovating its physical infrastructure as well as its research and education facilities.

Vision

To be a world class knowledge enterprise striving for academic & professional excellence in technical education, research and development and service to the industry and society

Objectives

- To provide world class quality technical education in engineering and science disciplines.
- To promote R&D in frontal areas of science and technology.
- To nurture innovative and creative abilities of students and faculty.
- To foster enterprising spirit among the students.
- To establish facilities for technology incubation and promote techno-entrepreneurship.

- To network with leading national and international institutions, R&D organization and professional bodies.
- To develop faculty competence to meet the challenges of rapidly changing technological environment.
- To promote industrial consultancy and sponsored R&D.
- To strengthen relations with the alumni and involve them in the development of the institute.
- To transfer the benefits of S&T advancement to the society

CERTIFICATE

DELHI COLLEGE OF ENGINEERING

(Govt. of National Capital Territory of Delhi)

BAWANA ROAD, DELHI - 110042

Date: _____

This is certified that the major project report entitled "**Design and Implementation of Semantic Search Tool With Automatic Document Classification and Summarization**" is the work of **Nitin Nizhawan** (University Roll No. 10191) a student of **Delhi College of Engineering**. This work was completed under my direct supervision and guidance and forms a part of the Master of Engineering (Computer Technology and Application) course and curriculum. He has completed his work with utmost sincerity and diligence.

The work embodied in this major project has not been submitted for the award of any other degree to the best of my knowledge

(Dr. DAYA GUPTA)

HOD & PROJECT GUIDE

(Dept. of Computer Engineering)

DELHI COLLEGE OF ENGINEERING

ACKNOWLEDGEMENT

This work is not a solo endeavour but rather the amalgamate consequence of contribution from various people and sources. Therefore, it would be discourteous to present it without acknowledging their valuable guidance.

It is distinct pleasure to express my deep sense of gratitude and indebtedness to my learned supervisors Dr. Daya Gupta, Mrs. Rama Hariharan (technical director NIC) for their invaluable guidance, encouragement and patient reviews. Their continuous inspiration only has made me complete this dissertation. All of them kept on boosting me time and again for putting an extra ounce of effort to realize this work.

I would also like to take this opportunity to present my sincere regards to my teachers Prof. D.Roy Choudhary, Mrs. Rajni Jindal, Dr. S. K. Saxena and Mr. Manoj Sethi for their support and encouragement.

Finally we are also thankful to my classmates for their unconditional support and motivation during this work. Last but not least, I special thanks to the crowd who are active in the field of Semantic Web.

(NITIN NIZHAWAN)

Master in Engineering

(Computer Technology & Application)

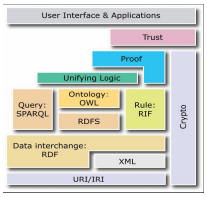
Dept. of Computer Engineering

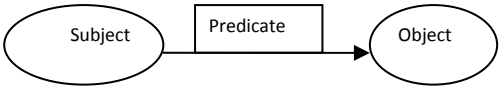
DELHI COLLEGE OF ENGINEERING

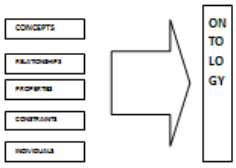
BRIEF CONTENTS

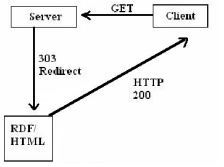
TABLE OF CONTENTS

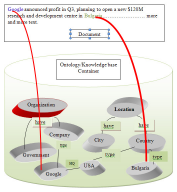
LIST OF FIGURES









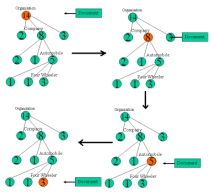


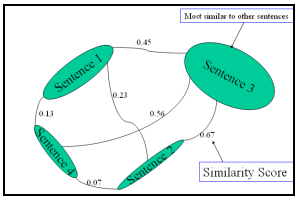
Word	Co-occurrents					
	a	friend	in	need	is	indeed
a	0	2	0	0	1	0
friend	2	0	1	0	0	1
in	0	1	0	1	0	0
need	0	0	1	0	1	0
is	1	0	0	1	0	0
indeed	0	1	0	0	0	0

Words	List of Documents
Word1	Doc1, Doc3, Doc 3, Doc 6…
Word2	Doc2, Doc10 , Doc4, Doc11…

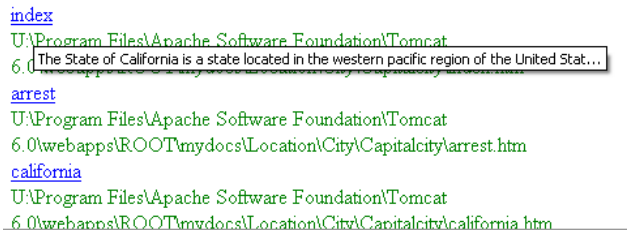
Documents	List of words
Doc1	Word1 Word5 Word9…
Doc2	Word2 Word13 Word14…

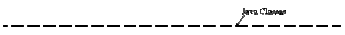
Fields	Doc1	Doc2
Filename	Doc1.txt	Doc2.txt
path	/doc1.txt	/doc2.txt
PERSON		gates,Sachin, bill
BUSINESSMAN		gates, bill
ELEC._DEVICE	gates, resistor	
SEMICON._DEVICE	gates	

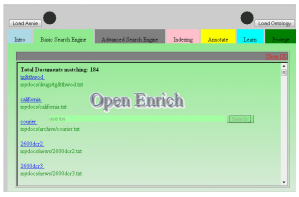


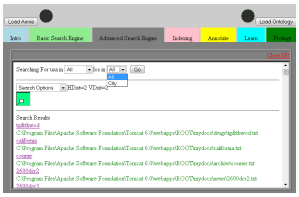


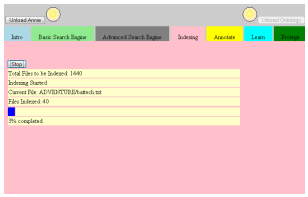
Fields	Doc1	Doc2
Filename	Doc1.txt	Doc2.txt
path	/doc1.txt	/doc2.txt
PERSON		gates,Sachin, bill
BUSINESSMAN		gates, bill
ELEC._DEVICE	gates, resistor	
SEMICON._DEVICE	gates	

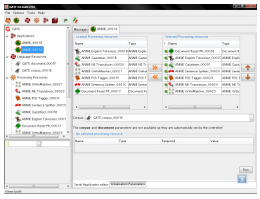


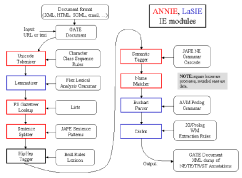












ABSTRACT

The web, once solely a repository for information, is evolving into a provider of services, such as e-commerce and searching. With this huge amount of information available the Semantic Web Technologies are one of the promising solutions for efficiently managing huge free form data in web documents. The Semantic Web aims to enrich the existing web with a layer of machine-interpretable metadata so that a computer program can draw conclusions predictably.

The HTML language has a major drawback. It defines markups that define the format in which a given text should be displayed but says nothing about the text. Thus internet user is presented with lot of irrelevant documents by present search engines, because they simply have no way to tell what is in the documents. More over they have no way to tell what a given word in the document means. These problems are solved by the Semantic Web Technology.

Enabling the web with semantic web technologies is one of the promising solutions for efficiently managing huge free form data in web documents. We present state of art methods available for semantically annotating documents, to index annotated documents, to provide semantically related search options, to provide context sensitive search, to present summary of document & to categorize documents according to "Named Entities" present in the document. We go on to build a tool for above services. This tool has been designed as a prototype for the plug-in of National Panchayat Portal at National Informatics Centre (NIC India) to make it as India's first semantic portal.

Chapter1

INTRODUCTION

"The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation."

- **Tim Berners-Lee**

1.1 Introduction to Semantic Web[1]

The Internet has provided us with a new dimension in terms of seeking and retrieving information for our various needs. Who would have thought about the vast amount of data that is currently available electronically ten years ago? When we look back and think about what made the Internet a success we think about physical networks, fast servers, and comfortable browsers, just to name a few. What one might not think about, a simple but important issue is the first version of HTML. This language allowed people to share their information in a simple but effective way. All of a sudden, people were able to define a HTML document and put their information piece on the Web.

The given language was sloppy and almost anybody with a small amount of knowledge about syntax or simple programming use could define a web page.

Even when language items such as end-tags or closing brackets were forgotten, the browser did the work and delivered the content without returning syntax errors. We believe this

to be a crucial point when considering the success story of the Internet: give the people a simple but effective tool with the freedom to provide their information.

Providing information is one thing, searching and retrieving information is at least as important. Early browsers or search engines offered the opportunity to search for specific keywords, mostly searching for strings. The user was prompted with results in a rather simple way and had to choose the required information manually. The more data were added to the Web, the harder the search for information became. The latest versions of search engines such as Google provide a far more advanced search based on statistical evidences or smart context comparisons and rank the results accordingly. However, the users still have to choose the information they are interested in more or less manually.

Being able to provide data in a rather unstructured or semi-structured way is part of the problems with automatic information retrieval. This is the situation behind the activities of the W3C concerning the Semantic Web. The W3C[2] defines the Semantic Web[3][4] on their Web page as:

"**The Semantic Web** *is the abstract representation of data on the World Wide Web, based on the RDF standards and other standards to be defined. It is being developed by the W3C, in collaboration with a large number of researchers and industrial partners."*

The same page contains a definition of the Semantic Web that is of similar importance. It states

"**The Semantic Web** *is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation*[5]."

These definitions indicate the Web of tomorrow. If data have a well-defined meaning, engines will be able to intelligently seek, retrieve, and integrate information and generate new knowledge to answer complex queries.

The retrieval and integration of information is the focus of this thesis.

Before going into detail we would like to share some creative ideas, which can be a vision of what we can expect from the Semantic Web.

1.1.1 Semantic Web Vision

Bernes-Lee et al. [6] already gave us an insight of what we should be able to do with the help of data and engines working in the Web. In addition, the following can help to see where researchers want to arrive in the future. These ideas can be distinguished into four groups:

Short-term: The following tasks are not far away from being solved or, are already solved to a certain extent.

Being able to reply on an email via telephone call: This requires communication abilities between a phone and an email client. Nowadays, the first solutions are available, however, vendors offer a complete solution with a phone and an email client that come in one package with more or less the same software. An example is the VoiceXML package from RoadNews3. The beauty of this point is that an arbitrary email client and an arbitrary phone can be used. The main subject is interoperability between address databases.

Meaningful browsing support: The idea behind this is that the browser is smart enough to detect the subject the user is looking for. If for instance, the user is looking for the program on television for a certain day on a web page, the browser could support the user by offering similar links to other web sites offering the same content.

Mid-term: These tasks are harder to solve and we believe that solutions will be available in the next few years.

Planning appointments with colleagues by integrating diaries: This is a problem already tackled by some researchers and the first solutions are available. Pages can be parsed to elicit relevant information and through reference to published ontologies reasoning support, it is

possible to provide user assistance. However, this task is not simple and many problems still have to be addressed. This task serves as one example of the ongoing Semantic Web Challenge

Context-aware applications: Ubiquitous computing might serve as another keyword in this direction. Context-awareness has to deal with mobile computing, reduction of data, and useful abstraction (e.g., digital maps in an unknown city on a PDA).

Giving restrictions for a trip and getting the schedule and the booking: The scenario behind this is giving a computer the constraints for a vacation/trip. An agent is then supposed to check all the information available on the Web, including the local travel agencies and make the booking accordingly. Besides some severe technical problems, such as technical interoperability between agencies, we also have to deal with digital signatures and trust for the actual booking at this point. First approaches include modern travel portals such as DanCenter4 where restrictions for a trip can be made and booking is also possible. This issue will be postponed for now.

Long-term: Tasks in this group are again more difficult and the solutions might emerge only in the next decade.

Information exchange between different devices: Suppose, we are surfing the Web and see some movies we are interested in which will be shown on television during the next few days. Theoretically, we are able to directly take this information and program our VCR (e.g., WebTV5).

Oral communication with the Semantic Web: So far, plain commands can be given via speech software to a computer. This task goes even further: here, we think about the discussions of issues rather than plain commands. We also anticipate inferences and interaction.

Lawn assistant: Use satellite and weather information from the Web, background garden knowledge issued to program your personal lawn assistant.

Never: Automatic fusion of large databases.

We can identify a number of difficult tasks that will most likely be difficult to solve. The automatic fusion of large databases is an example for this. On the other hand, we have already

seen some solutions (or partly solutions) to the tasks that are grouped into short- and mid-term problems (e.g., integrating diaries). The following research topics can be identified with regard to theses ideas.

1.1.2 Search on the Web

Seeking information on the Web is widely used and will become more important as the Web grows. Nowadays, search engines browse through the Web seeking given terms within web pages or text documents without using ontologies.

Traditional search engines such as Yahoo are based on full-text search.

These search engines are seeking documents, which contain certain terms. In order to give a more specific query, the user is often able to connect numerous terms with logical connectors such as AND, OR or NOT. The program extracts the text found from the documents and delivers the answer (usually a link to the found document) to the user. However, these search engines also use algorithms that are based on indexing for optimization purposes. The search engine then uses this index for seeking the answer. Yahoo has shown that this kind of search can be sufficient if the user knows what they are looking for.

A clear disadvantage here is the fact that these search engines only search textual documents. Also, they have problems with synonyms, homonyms or a mistake while typing. These engines usually provide a huge amount of results that fulfill the requirement of the query, however, most of the results are not what the user intended.

Another type of search is the similarity-based search used in search engines such as Google. The engine is looking for documents, which contain text that is similar to a given text. This given text could be formulated by the user who is seeking the information or can be a document itself. The similarity is analyzed by the words used in the query and the evaluated documents. The engine usually uses homonyms and synonyms in order to get better results.

The method extracts the text corpus out of the document and reduces it to a number of terms. A distance measure assigns the similarity to a numerical value between 0 and 1, where the similarity is the determined by the number of corresponding terms. The advantage of this kind of search is that there is no empty set of results and the results are ranked. A disadvantage is that only text documents can be used. Also, the similarity is based in given words and sometimes it is hard to find appropriate words for the search.

The main problem in these kinds of search is that the amount of results is numerous. Also, most of the results are not accurate enough. The user has to know the terms they are looking for and cannot search within documents other that textual-based files and web pages. The reason for this is that uninformed search methods do not use background knowledge about certain domains.

Intelligent search methods take this into account and use additional knowledge to get better results. However, this requires a certain extent of modeling for the knowledge. The given documents are annotated with extra knowledge (metadata). The search can then be extended by search about the annotated metadata. This background knowledge can be employed for the formulation of the query by using ontologies and inference mechanisms. Also, the user can use this extra knowledge to generate abstract queries such as “all reports of the department X”. The reports can be project reports, reports about important meetings, annual reports of the department etc. With ordinary search engines the user would have to ask more than once.

Intelligent search methods also include the classical way of search. The user will get more sophisticated results if he takes advantage of the additional knowledge. If the users do not know the exact terms they are looking for, they can also take advantage of the extra knowledge by using inference mechanisms of the ontology. However, this requires that the knowledge is formulated in a certain way and inference rules need to be available. The Semantic Web provides information with a well-defined meaning, and in the following we will use the term “search” for “intelligent search”.

We have mentioned how intelligent search can help us to get better results. We have also explained that ontologies are the key to this. Seeking information with ontologies adds a new

feature to the search process: we are able to use inference mechanisms in order to derive new knowledge. The search would even be more efficient if we would be able to integrate information from data sources. Integration in this context means that heterogeneous information sources can be accessed and processed despite different data types, structures, and even semantics.

1.2 Proposed Research Work

National Informatics Centre (NIC) has a project National Panchayat portal. The portal's aim is to collect all data and information from all Village Panchayats in India. It collects the data from more than 200 thousand Village Panchayats in India. People upload data on the web portal in the form of documents. Administrator analyses the files uploaded by the people of Village Panchayats. People also submit information about the content they are submitting like the name of author, its main topic, Title etc. These files are stored and indexed. A search engine uses this metadata to index documents and provide search results to users.

The metadata that users upload is not sufficient for organising the huge amount of data generated by all over India. Therefore need for the generation of semantic metadata and using this metadata to provide for better search. And also the use of other natural language processing techniques to improve the retrieving efficiency of the system.. Therefore we chose to make the search engine of the portal a semantic search engine, which will allow users to easily find the information they require, and I chose **following** topic of research for my thesis work.

"Design and Implementation of Semantic Search Tool with Automatic Document Classification and Summarization."

1.3 Related Work

A lot of work is being done in the field of semantic web. A number of approaches are being presented. We came across one such paper [7]. This paper presents and alternative way of creating semantic web. It calls such web, A Soft Semantic Web[7]. This paper presents an

approach to creating semantic web without using ontology. It proposes to use statistical approach by taking joint frequency of the keywords as the parameter. It suggests that such joint keyword frequency can represent concepts. Another paper by Rada F. Mihalcea and Silvana I. Mihalcea [8] propose the use of WordNet[9] to enhance the information in the documents which can be used to increase the efficiency of information retrieval. In the Paper by Iraklis Varlamis et al [10] propose the use link semantics to enhance the web content management. They propose to analyse the semantics of a link in addition to link structure analysis done by major search engines.

1.4 Proposed System

We found that to meet the challenges of searching a new tool with all the functionalities integrated together was required. Therefore we developed the tool with required techniques and algorithms for the project. The integrated tool performs following tasks.

- **Summarizing Documents.**
- **Classifying Annotated Documents.**
- **Indexing Annotated Documents.**
- **Providing Semantic Search**

The tool that is to be described in this thesis performs all above function and it was design for the plug-in of *National Panchayat Portal* at National Informatics Centre (NIC India) to make it as ***India's first semantic portal.***

1.4.1 Document Summarization

The main aim of this project is make data and information easily available to the users. One simple solution that does not use ontology and annotation is to summarize the document. The summarization of document allows user to know what is in the document before even reading the complete document. This summarization should be automatic because of sheer size of the data and number of documents. The summary of the documents can then be listed along side the search results and reducing the time user will require to reach the required document.

1.4.2 Document Classification

This is another technique that would prove useful in achieving the aim mentioned above. The idea is to classify document in categories depending upon the type of type of information they may contain. This enables user to read only those documents that are closest to the content s/he is looking for. The types into which document should be classified are described in the ontology. We devise a algorithm that classifies the document into classes described in the ontology. The algorithm ensures that the hierarchy described in the ontology is maintained after the classification also, which allows users to choose the class depending upon the generality of the concept they are searching for.

1.4.3 Indexing

Index is a common data structure used to speed up the searching process. In this application also the indexing of documents war required to speed up the search later performed by the searcher. But in this case the documents were annotated, they contain semantic information. Therefore an index strategy was required that would effectively make use of this additional information which can later be utilized efficiently by the search engine.

1.4.4 Semantic Search

All above features are background processes; the search is what users do. It provides a basic search interface to the users. It provides semantic search by analysing the query of the user and taking action depending upon how that query relates to the ontology. It allows user to give semantic information about the keyword s/he is searching for and also provides more information that is semantically related to the user query. This will allow user to search in a more comprehensive manner. This feature also combines above all features and presents them to user so that they are able to search faster.

1.5 Organization of Thesis

This thesis is organized as follows

Chapter 1 gives an introduction to semantic web idea and an introduction to this thesis. It introduces to the main aim and motivation of the thesis.

Chapter 2 describes the semantic web fundamentals with the brief description of semantic web, RDF/RDFS[11], Ontologies[12] and its language OWL, Annotation with semantic annotation, URL's And Semantic web architecture, information and non-information resources, basic idea of HTTP and XML in brief.

Chapter 3 describes the theory of document summarization and classification. It starts with an introduction to natural language processing. In its second section it describes document classification technique used in the project. Last section on summarization describes the summarization introduction, history and the word space model.

Chapter 4 throws light on searching feature of the semantic web. It describes types of searching and problems faced by current searching method and It also describes that how indexing of document should be done to support semantic search feature.

Chapter 5 Describes in detail the architecture of the tool that we have developed. It explains different modules, data structures and various algorithms used for semantic search, indexing, summarization of documents and classification.

Chapter 7 covers the implementation details with the block diagram of the tools showing how various modules communicate with each other. It also shows the state of art GUI we have developed for the tool.

Chapter 8 Culminates the project thesis with Conclusion. It also throws light on the future possibilities in the field on semantic web.

Publications from Thesis gives list of research papers that we wrote during our research. It also gives their status and the conferences and journals in which they have been accepted or communicated.

References give a list of various research papers, websites, books that we have gone through during the research

Appendices gives an overview of the tools (Protege[13],GATE[14], Lucene[15]) used in this project.

Chapter 2

SEMANTIC WEB TECHNOLOGIES

**"If I have seen farther than others,
it is because I was standing on
the shoulders of giants"**

- Sir Isaac Newton (1642-1727)

This section provides a general introduction to the Semantic Web and its associated technologies. The main components of the Semantic Web including RDF, OWL and URIs are all discussed. An introduction is given to Open eNRICH a software tool for creation of web portals developed by National Informatics centre (India), Ontology and Annotation.

2.1 Overview

The World Wide Web as it exists today consists of many millions of documents accessible through The Internet. Documents can consist of text, audio, video or indeed can be of any format that can be read on a computer. These documents are mainly produced for human consumption, i.e. interaction by a user or simply text that can be read. Therefore the majority of the documents on the Web are meant to be understood by humans and used by humans.

Although the Web has provided a useful and sometimes invaluable resource, there is a potential to make the Web even more useful than it is today. A new level of functionality could be added to the Web if, instead of being human oriented, the web or documents on the web could be understood by machines as well as humans. The need to make The Web machine understandable sparked the beginning of The Semantic Web [4].

The Semantic Web is an effort, to effectively organize all of the knowledge on the Web, Make it machine understandable using a common set of standards, make it universally available to different forms of devices; and provide services that will help us achieve tasks that, at the moment, require some amount of time and effort.

To give us an idea of what The Semantic Web would look like a futuristic scenario was presented in which electronic intelligent agents communicate with each other using the Semantic Web to automatically create doctors appointments and buy medication on prescription (BernersLee 2001). There are many such scenarios in which The Semantic Web could fully automate our everyday tasks. When somebody would like to go on holiday abroad, The Semantic Web would enable the flights, travel, accommodation and entertainment to be booked automatically. When someone would need to replace a part of their computer, they would only need to describe that component and then all of the manufacturers and sellers would automatically be contacted and the best priced component would be ordered. This is obviously a very challenging goal and requires a lot of effort and research in order to make it a reality. The approach so far has been to build each part of the 'cake' (see next section) piece by piece until eventually all the different areas of research converge. During the last few

Year's significant progress has been made in laying the foundation of The Semantic Web which has now culminated in W3C approved specifications for knowledge representation (RDF) [11] and ontology definition (OWL). With these specifications in place applications are now being developed that demonstrate the potential power of The Semantic Web (Shadbolt et al. 2004), (Karger et al. 2005), (Noy et al. 2000).

Semantic Web is being developed in order to overcome the following main limitations of the current Web:

- No structured information - the majority of web documents has a lack of structure regarding the representation of information;

- Ambiguity of web content - the inexistence or poor aggregation of information to specific contexts;

- Inadequacy for automatic information transfers;

- Increasing difficulty to locate, access, present and maintain an online trustful content for an increasing number of users;

- Inexistence of universal mechanisms able to provide the capacity to machines to understand the information.

Semantic Web goal is not to make computers understand the human language, but to define an universal model for the information expression and a set of inference rules that machines can easily use in order to process and to relate the information as if they really understand it Though, as the current Web allowed the sharing of documents among previously incompatible computers, the Semantic Web intends to go beyond, allowing stovepipe systems, hardwired computers and other devices to share contents embedded in different documents. The main advantages of Semantic Web can be grouped into several features as follows:

- Allows the description and spread of the current Web, adding to it a concept layer.

- Allows machine-readable, interpretable and editable web content.

- Offers a way to enable semantic annotations that could be easily organized and found.

- Enhances search mechanisms with the use of Ontologies.

- Enables software agents to carry automatically sophisticated tasks - through the use of smart data.

- Allows better communication between platform independent software agents.

- Enables the use of levels of trust for information.

2.2 Knowledge Representation and Management

2.2.1 RDF/RDFS

The Semantic Web is based around a core set of standards that have been developed by the W3C (World Wide Web Consortium). This is commonly described as the Semantic Web Layer Cake', (BernersLee, T., 2007) and is shown in Figure

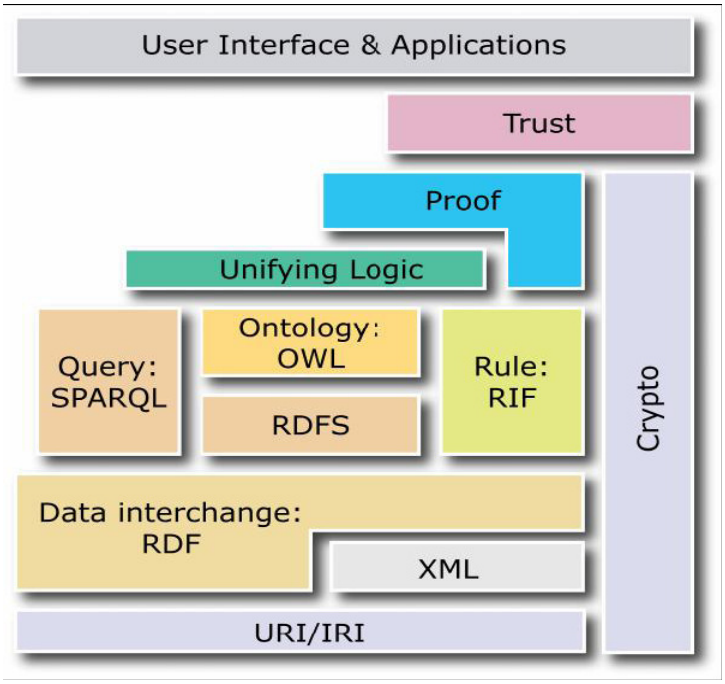


Figure 2.1 Semantic Web Layer Cake

The most important language shown in figure 2.1 is RDF [11] and its associated vocabulary description framework RDF Schema. The Resource Description Framework is a standard that 'provides interoperability between applications that exchange machine understandable information on the Web' (Lassila & Swick 2002). The specification enables things in the real world to be described in the same way that humans use sentences to describe things.

The basic description takes the form:

<subject> <predicate> <object>

A subject takes the form of a URI (Uniform Resource Identifier), which is a way of identifying any object in the world. It looks the same as a URL but the address which is given may not be accessible or may not even exist on the Web, instead it is used as a way of linking objects to some definite identifier. The predicate part, which is also a URI, describes the property of the subject. The object is the thing that is doing the predicate. For example the sentence, 'The Delhi Engineering College has a student called Nitin Nizhawan', contains a subject, 'The Delhi engineering College', a predicate, 'student' and an object, 'Nitin Nizhawan'. Such a sentence is easy to translate into basic RDF/XML syntax and is expressed as:

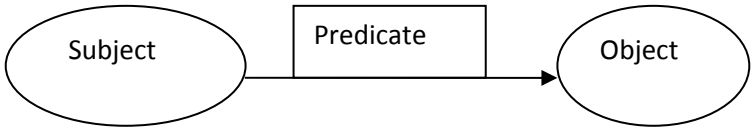


Figure 2.2 RDF/XML syntax representations

```
<rdf:RDF>
<rdf:Description about="http://www.dce.ac.in">
<akt:hasstudent>Nitin Nizhawan</akt:hasstudent>
</rdf:Description>
</rdf:RDF>
```

This assumes that the entity The Delhi Engineering College is represented by the URI, "http://www.dce.ac.in". The '*akt:*' refers to a namespace prefix that is declared in a namespace declaration at the beginning of the RDF document. The declaration contains the identifier where

the schema to describe the property ‘has student’ can be found. RDF also contains many other features such as containers and ways to make statements about statements. The RDF specification can express a number of different properties of entities and relationships between those entities. The RDFS specification builds on RDF to allow the creation of classes and subclasses to be used in much the same way as they are used in taxonomic classifications. Together, RDF and RDFS have become the accepted way of describing knowledge of specific domains and information about real world entities.

2.2.2 Ontologies and OWL

Ontologies[12] can play a crucial role in enabling Web-based knowledge processing, sharing, and reuse between applications. Generally defined as shared formal conceptualizations of particular domains, ontologies provide

a common understanding of topics that can be communicated between people and application systems.

There are some disputes in knowledge representation as to what exactly ‘Ontology' is We consider 'Ontology' is built which includes a collection of domain-specific concepts, and it is a system description which should include categories, relationships and constraints between them. Ontology, Thesaurus and Taxonomy are method of the information organization.

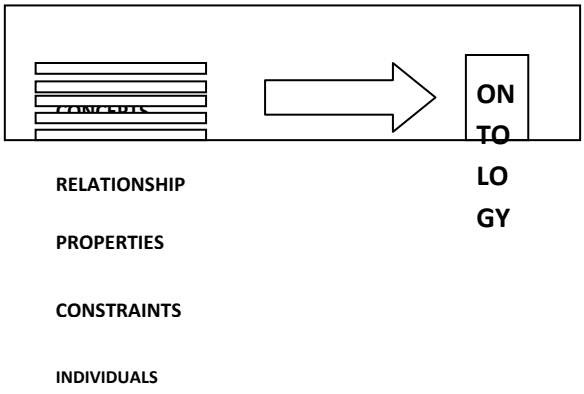


Figure 2.3 Ontology Description

In the design of ontology[16][17], the basic element is a 'category', which meaning its domain. Each category should have 'individual' and 'instance'. Their basic relation is 'inheritance' and 'implement'. In order to depict them, 'Slot' could be used. At the same time, slot is divided into attribute and relationship and limited with facet.

In order for a machine to be able to understand objects in the real world, there needs to be a way to represent and classify these objects. We define and associate objects in the real world by the knowledge that we have about them. Ontology is a means of being able to store knowledge about a particular subject area or for multiple subjects within some specific domain, and process it in some way.

These descriptions are rather vague and there is no way of showing how ontology can be represented in a form that a computer can read and process. Previous research has led to the development of languages to represent ontologies such as description logics, Frame Logic (Kifer et al. 1995) and Knowledge Interchange Format (KIF) (Genesereth et al. 1992). There have also been full scale applications made from these languages that are in use today (Fensel, 2001).

The main use of ontology is to describe a domain using an appropriate vocabulary and define relationships between the terms taken from the vocabulary. One of the most natural ways for humans to classify objects is to use hierarchies. The RDFS [11] specification reflects this by using classes and subclasses and has the ability to define properties that can be given to an instance of a class. However, in order to establish more complex relations, such as cardinality constraints, optional constraints or disjointness, a more complex syntax is required. This has led to the development of OWL (Web Ontology Language) which has now become the standard for ontology

definition. This language defines syntax for describing classes and properties as well as more complex relationships. The OWL language ‘is designed for use by applications that need to

process the content of information instead of just presenting information to humans' (McGuiness & Harmelen 2004).

There are three types or 'flavours' of OWL: OWL Lite, OWL DL and OWL Full. OWL Lite is a subset of OWL DL and OWL DL is a subset of OWL Full. The main difference between OWL full and the other two flavors of OWL is that in OWL Full instances of a class can be classes, where as in OWL Lite and OWL DL, instances of a class must be individuals. In practice, this means that working with OWL Full is generally too complex for a logic reasoner to use for logical deduction, but OWL DL is both complete and decidable and is therefore easier to reason over and use.

2.2.3 Annotation

Annotation', in contemporary English, according to WordNet [18], have two meanings:

1. Note, annotation, and notation: a comment (usually added to a text).
2. Annotation, annotating -- the act of adding notes.

In linguistics (and particularly in computational linguistics) an annotation is considered a formal note added to a specific part of the text. There are number of alternative approaches regarding the organization, structuring, and preservation of annotations[19]. For instance, all the markup languages (HTML, SGML, XML, etc.) can be considered schemata for embedded or in-line annotation. On the contrary, open hypermedia systems use stand-off annotation models where annotations are kept detached, i.e. non-embedded in the content.

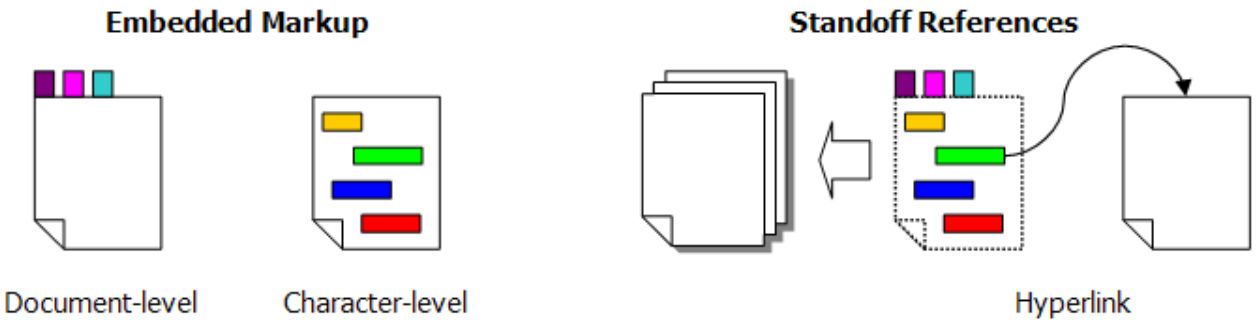


Figure 2.4 Embedded Vs Standoff Mark up

We annotate data all the time: when we read a paragraph, and mark "great!" in the margin, that is an annotation. When our text editor underlines a misspelled word, which is also an annotation. Annotations add some information to some other information; to annotate means "to make notes or comments".

Another way to view annotations is metaphorically: URIs are the "atoms" of the Semantic Web and semantic annotations are the "molecules". The Semantic Web is about shared terminology, achieved through consistent use of URIs. Annotations create a relationship between URIs and build up a network of data.

2.2.4 Semantic Annotation

Semantically annotated [20] documents play an important role in semantic web by using and inserting tags according to the meaning of the word in context. The semantic meanings and interpretations of keywords are bound to that domain and therefore the retrieval is likely to be more efficient.

2.2.5 URLs and Semantic Web Architecture

The base of the layer cake shown in figure is URIs and IRIs. Uniform Resource Identifiers and their corresponding Internationalized Resource Identifiers are central to the design and use of the Semantic Web. A URI provides a common syntax that can be used to name or identify any entity in the world.

For example, the URI "http://id.dce.ac.in/person/6751" is an identifier used to refer to the person 'Nitin Nizhawan' that has been provided by DCE. This URI uses the http naming scheme although other schemes are available such as urn, dav, file and ftp. The universality of the URI is a fundamental part of Web architecture. The idea of using URIs is that they should be something into which any system's identifiers can be mapped. Providing a name for different entities and resources means that descriptions of objects can be easily made and published so that other agents or systems can combine knowledge from different sources. This relies on the assumption that every object will have a unique URI. However, anyone who wants to make statements or give knowledge about an object can introduce their own URI to refer to that object. Thus, there maybe many URIs that refer to the same real world entity. The relationship between a URI and a URL is a subtle one. In the early days of the Web it was thought that the URL was a subclass or special type of URI (W3C, 2001). The contemporary view is that a URL is only an informal concept that provides the address of a location to access a resource. Therefore it can be said that a URI that uses the 'http' scheme is also a URL.

2.2.6 Information and Non-Information Resources

The current Semantic Web architecture divides entities and objects that have URIs into two categories: information resources and non-information resources. Non-Information resources are those entities that cannot be represented as a byte stream or serialized into a character format. This category of objects covers things such as cars, people, places, books and concepts that exist in the real world and not on the Web. The category of information resources on the other hand, consist of resources that can be

accessed on the Web such as HTML pages, JPEG images, PDF documents and RDF descriptions. Information resources can be used to describe non-information resources in a way that can be accessed using HTTP. The two categories of resource have been made in order to be able to represent real world entities on the Web. The URI of an object cannot also contain the description of that object.

In order to use URIs and make statements about the objects that the URIs refers to, a mechanism had to be constructed so that the URI and description stay separated. The way that this is accomplished is to introduce an HTTP 303 redirection when the URI of a resource is resolved or dereferenced on the Web. For example, when the URI for 'Nitin Nizhawan ' is put into a Web browser, the browser will issue an HTTP GET to the server, who in turn will redirect the browser, via 303, to a description of the URI in the format that was requested by the browser. Each description of the URI will itself have a URI so that the distinction between information and noninformation resources is preserved. The architecture is illustrated in Figure with the URI for 'Nitin Nizhawan' used together with the URIs for a text/html and rdf+xml description of the URI.

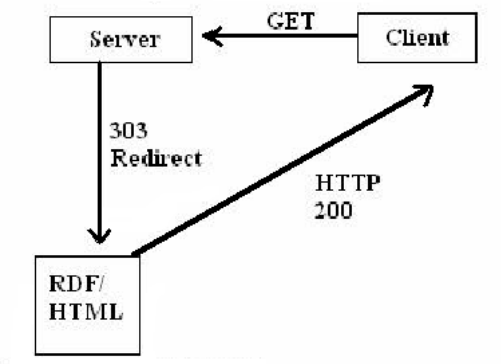


Figure 2.5 SW Architechture for non- information sources

2.2.7 Introduction to HTTP

The Hypertext Transfer Protocol (HTTP) is a generic and stateless protocol for distributed, collaborative, hypermedia information systems. It allows performing operations on resources which are identified by URIs. It has been widely used for more than one decade and now comes to version 1.1. It has four main methods:

a. Get means get the information that is identified by the request URI. Usually it is the action we take when browsing sites and clicking hyperlinks, we ask the web server for the resources that is identified by Request-URLs.

b. Post means make a request to the web server so that the server accepts the resources encapsulated in the request, which will be the new subordinate of the resource identified by the Request-URI in the Request-Line. Usually it is what we do when filling and sending the HTML form to the server to buy something like CDs, books etc. -- making a request of the server.

c. Put means make a request that sends updated information about a resource if the resource identified by the Request-URI exists, otherwise the URI will be regarded as a new resource. The main difference between the POST and PUT requests lies in the different meaning of the Request-URI. In a POST request, the URI is to identify the resource that will handle the enclosed entity. As for the PUT request, the user agent knows what URI is its aim and the web server cannot redirect the request to other resources. Unfortunately most web browsers don't implement this functionality, which makes the Web, to some extent, a one-way medium.

d. Head is similar to GET except that the server don't return a message-body in the response. The benefit of this method is that we can get meta-information about the entity implied by the request without transferring the entity-body itself. We can use this method to check if hypertext

links are valid, or if the content is modified recently .By using HTTP, Semantic Web can benefit all these functionalities for free. In addition, almost all HTTP servers and clients support all these features.

2.2.8 XML

Extensible Markup Language (XML) is a subset of SGML (the Standard Generalized Markup Language), i.e. it is totally compatible with SGML. But it is simple and flexible. It's original aim to tackle the problems of large-scale electronic publishing. However, it is also very important in data exchange on the Web. Despite its name, XML is not a markup language but a set of rules to build markup languages.

Markup language

"Markup is information added to a document that enhances its meaning in certain ways, in that it identifies the parts and how they relate to each other." - Erik T. Ray, Markup language is kind of mechanism organizing the document with a set of symbols, e.g. this article is labeled with different fonts for headings. Markup use similar methods to achieve its aims. Markup is important to implement machine-readable documents since a program need to treat different part of a document individually.

Namespaces

We can expand our vocabulary by namespaces which are groups of element and attribute names. Suppose, if you want to include a symbol encoded in another markup language in an XML document, you can declare the namespace that the symbol belongs to. In addition, we can avoid the situation that two XML objects in different namespaces with the same name have different meaning by the feature of namespaces. The solution is to assign a prefix that indicates which namespace each element or attribute comes from. The syntax is shown below:

ns-prefix*: *local-name

XML Schemas

XML itself does not do anything, i.e., it is just structure and store information. But if we need a program to process the XML document, there must be some constraints on sequence of tags, nesting of tags, required elements and attributes, data types for elements and attributes, default and fix values for elements and attributes and so on. XML Schema is an XML based alternative to Document Type Definition (DTD) . There are some features of XML Schemas that overweigh DTD:

a.) XML Schemas support data types, which brings a lot of benefits, e.g. easy to validate the correctness of data, easy to work with databases, easy to convert data between different types.

b.) XML Schemas have the same syntax as XML so that it can benefit all features of XML.

c.) XML Schemas secure data communication since it can describe the data in a machine-understandable way.

d.) XML Schemas are extensible because they are actually XML and then share this feature of XML.

e.) Well-formed is not enough since it also may contain some semantic confusion which can be caught by XML Schemas.

Chapter 3

DOCUMENT SUMMARIZATION AND CLASSIFICATION

"No, I'm not interested in developing a powerful brain. All I'm after is just a mediocre brain, something like the President of the American Telephone and Telegraph Company (AT&T)"

- Alan Turing (1912 – 1954)

3.1 Introduction to NLP

Natural language processing provides a potential means of gaining access to the information inherent in the large amount of text made available through the Internet. In today web environment where terabytes of data generated everyday, it is hard to read all the data generated everyday manually. So, for this some kind of natural language processing is necessary for computer to process this data.

Natural Language Processing (NLP) is an area of research and application that explores how computers can be used to understand and manipulate natural language text or speech to do useful things. NLP[21] researchers aim to gather knowledge on how human beings understand and use language so that appropriate tools and techniques can be developed to make computer systems understand and manipulate natural languages to perform the desired tasks. The foundations of NLP lie in a number of disciplines, viz. computer and information sciences, linguistics, mathematics, electrical and electronic engineering, artificial intelligence and robotics,

psychology, etc. Applications of NLP include a number of fields of studies, such as machine translation, natural language text processing and summarization, user interfaces, multilingual and cross language information retrieval (CLIR), speech recognition, artificial intelligence and expert systems, and so on.

At the core of any NLP task there is the important issue of natural language understanding. The process of building computer programs that understand natural language involves three major problems: the first one relates to the thought process, the second one to the representation and meaning of the linguistic input, and the third one to the world knowledge. Thus, an NLP system may begin at the word level – to determine the morphological structure, nature (such as part-of-speech, meaning) etc. of the word – and then may move on to the sentence level – to determine the word order, grammar, meaning of the entire sentence, etc.— and then to the context and the overall environment or domain. A given word or a sentence may have a specific meaning or connotation in a given context or domain, and may be related to many other words and/or sentences in the given context.

Liddy (1998) and Feldman (1999) suggest that in order to understand natural languages, it is important to be able to distinguish among the following seven interdependent levels, that people use to extract meaning from text or spoken languages:

- phonetic or phonological level that deals with pronunciation
- morphological level that deals with the smallest parts of words, that carry a meaning, and suffixes and prefixes
- lexical level that deals with lexical meaning of words and parts of speech analyses
- syntactic level that deals with grammar and structure of sentences
- semantic level that deals with the meaning of words and sentences
- discourse level that deals with the structure of different kinds of text using document structures and
- Pragmatic level that deals with the knowledge that comes from the outside world, i.e., from outside the contents of the document.

A natural language processing system may involve all or some of these levels of analysis.

3.2 Document Classification

3.2.1 Introduction

Document classification/categorization is a problem in information science. The task is to assign an electronic document to one or more categories, based on its contents. Document classification tasks can be divided into two sorts: supervised document classification where some external mechanism (such as human feedback) provides information on the correct classification for documents, and unsupervised document classification, where the classification must be done entirely without reference to external information. There is also a semi-supervised document classification, where parts of the documents are labelled by the external mechanism.

Document classification techniques include:

- naive Bayes classifier
- tf-idf
- latent semantic indexing
- support vector machines
- artificial neural network
- kNN
- decision trees, such as ID3
- Concept Mining

Unsupervised Document Classification

Document clustering or unsupervised document classification has been used to enhance information retrieval. This is based on the clustering hypothesis, which states that documents having similar contents are also relevant to the same query. A fixed collection of text is clustered into groups or clusters that have similar contents. The similarity between documents is usually measured with the associative coefficients from the vector space model, e.g., the cosine

coefficient. Hierarchical clustering algorithms have been primarily been used in document clustering. The single link method has mostly been used, as it is computationally feasible, but the complete link method seems to be the most effective but is very computationally demanding.

Other methods than document vector similarity have been used for clustering. Neural models have been implemented for unsupervised document clustering.

The long computation time has always been the problem when using document clustering on-line. More recently fast algorithms for clustering have been introduced to use for browsing through collection when the user has little information about the collection and wants to brows for topics. Suffix Tree Clustering is new clustering method, which creates clusters based on phrases shared between documents, works fast and intended for Web document clustering. Different projections techniques, LSI and truncation, have been investigate to speed up the distance calculations of clustering. An interesting application of clustering is topic clustering, i.e. clustering documents returned from a specific query, using -means clustering.

Supervised Document Classification

Pattern recognition and machine learning has also been applied to document classification. As before the term frequency is used as feature. A number of classifiers have been used to classify documents. An example of these classifiers are neural networks, support vector machines , genetic programming, Kohonen type self-organizing maps, hierarchically organized neural network built up from a number of independent self-organizing maps, fuzzy -means, hierarchical Bayesian clustering , Bayesian network classifier , and naive Bayes classifier .

Some of these classifiers can be used with unsupervised learning, i.e., unlabeled documents, but the accuracy of a classifier can be enhanced by using a small set of labelled documents. The aim is to use a classifier which need small amount of manually classified documents to be generalized.

The use of semi-supervised machine learning has emerged recently. The learning scheme lies somewhere between supervised and unsupervised, where the class information are learned from the labelled data and the structure of the data from the unlabeled data.

3.2.2 Proposed Technique

The technique used is to first semantically annotate the documents. Semantic Annotation [19] is about assigning to the entities in the text links to their semantic descriptions. This sort of metadata provides both class and instance information about the entities. It is a matter of terminology whether these annotations should be called “semantic”, “entity” or some other way. To the best of our knowledge there is no well-established term for this task; neither there is a well-established meaning for “semantic annotation”. What is more important, the automatic semantic annotations enable many new applications: highlighting, indexing and retrieval, categorization, generation of more advanced metadata, smooth traversal between unstructured text and available relevant knowledge. Semantic annotation is applicable for any sort of text – web pages, regular (non-web) documents, text fields in databases, etc.

Noun phrasing is considered to be an important NLP technique used in information retrieval. One of the major goals of noun phrasing research is to investigate the possibility of combining traditional keyword and syntactic approaches with semantic approaches to text processing in order to improve the quality of information retrieval. We use here this feature of NLP to populate our ontology and for semantically annotating of web documents. The nouns and proper nouns identified by using the noun phrasing technique are the candidate entities for populating the ontology and for annotating the documents. The complete approach for annotating and ontology population is described in the section given below.

Manual annotation is difficult, time consuming and expensive. Convincing millions of users to annotate documents for the Semantic Web is difficult and requires a world-wide action of uncertain outcome .For our tool to perform automatic semantic annotation in the documents we have used GATE[14] **General Architecture for Text Engineering** or **GATE** is a Java software toolkit originally developed at the University of Sheffield since 1995 and now used worldwide by a wide community of scientists, companies, teachers and students for all sorts of natural language processing tasks, including information extraction in many languages.

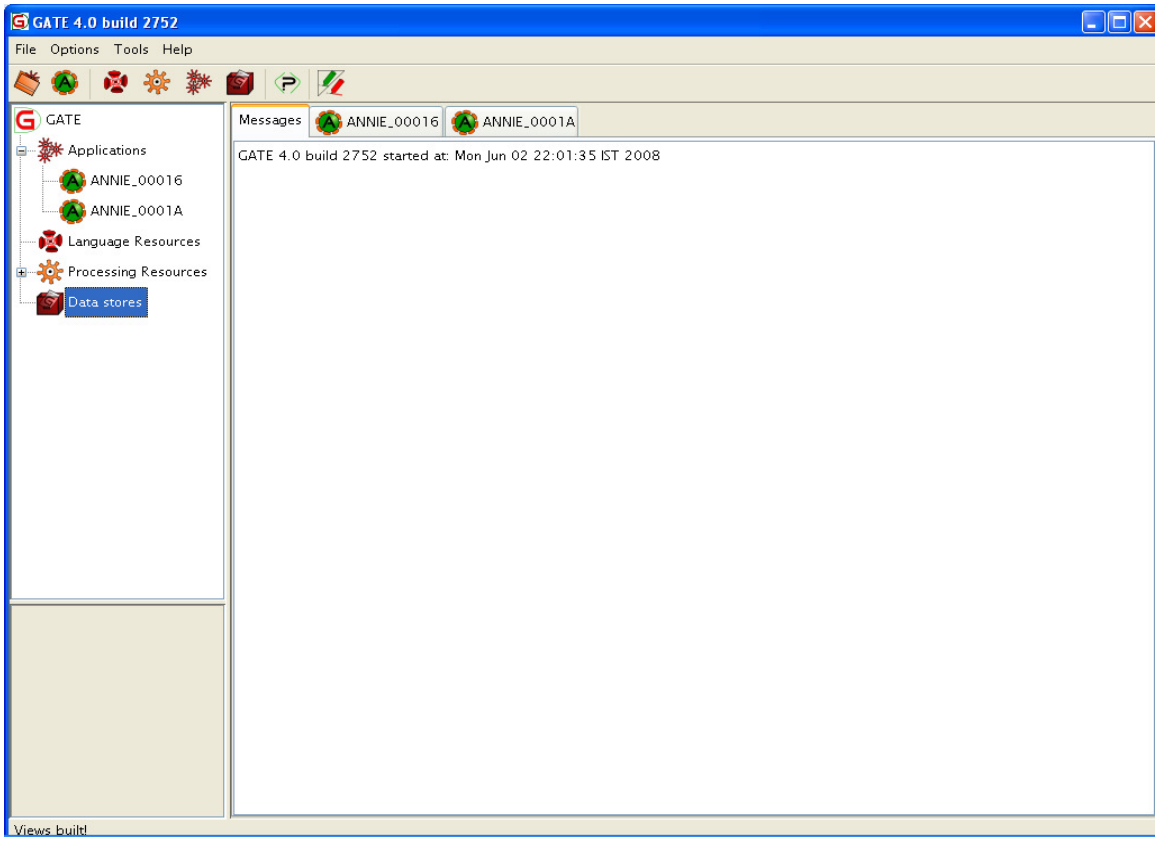


Figure 3.1 GATE GUI

The GATE tool provides a list of all the named entities in the document. The type of these named entities is first determined using the ontology and then it is encoded in the document this process of semantic annotation is shown in figure below.

After the document is annotated it is analysed to find out the type of entities is mainly contains. The algorithm used for this is described in the next chapter. The criteria used for classification should be such that if a document is classified in a class then it should also belong to its super lass according to same criteria. This is done to maintain the taxonomic relation ship between the classes defined in the ontology. For example in the figure above if the document has been classified as of lass ‘country’ then it should also belong to lass ‘Location’ using sae criteria Such an automatic classification allows users to focus easily on what they want classification

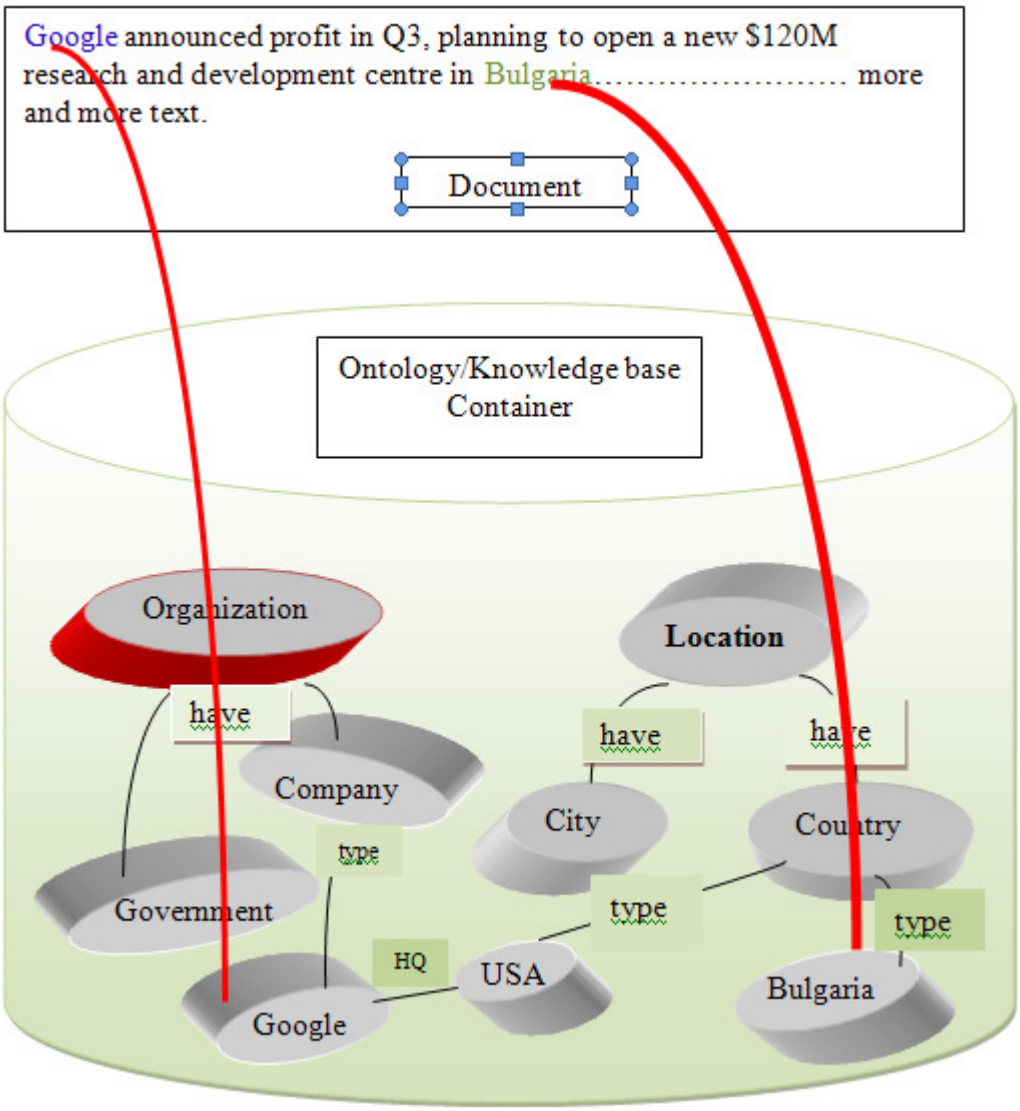


Figure 3.2 Semantic Annotation

3.3 Document Summarization

3.3.1 Introduction

Automatic Text Summarization is an important and challenging area of Natural Language Processing. The task of a text summarizer is to produce a synopsis of any document or a set of documents submitted to it.

Summaries differ in several ways. A summary can be an *extract* i.e. certain portions (sentences or phrases) of the text is lifted and reproduced verbatim, whereas producing an *abstract* involves breaking down of the text into a number of different key ideas, fusion of specific ideas to get more general ones, and then generation of new sentences dealing with these new general ideas. A summary can be of a single document or multiple documents, *generic* (author's perspective) or *query oriented* (user specific), *indicative* (using keywords indicating the central topics) or *informative* (content laden).

3.3.2 Previous Work in Extractive Text Summarization

Various methods have been proposed to achieve extractive summarization. Most of them are based on scoring of the sentences. **Maximal Marginal Relevance** scores the sentences according to their relevance to the query, Mutual Reinforcement Principle for Summary generation uses clustering of sentences to score them according to how close they are to the central theme. **QR decomposition method** scores the sentences using column pivoting. The sentences can also be scored by certain predefined features. These features may include linguistic features and statistical features, such as location, rhetorical structure, presence or absence of certain syntactic features and presence of proper names, and statistical measures of term prominence.

Rough set based extractive summarization has been proposed that aims at selecting important sentences from a given text using rough sets, which has been traditionally used to discover patterns hidden in data.

Methods using similarity between sentences and measures of prominence of certain semantic concepts and relationships to generate an extractive summary have also been proposed.

Some commercially available extractive summarizers like Copernic and Word summarizers use certain statistical algorithms to create a list of important concepts and hence generate a summary.

The paper [22] proposes to achieve extractive summarization as a three-step process:

1. Mapping the words and sentences onto a semantic space (Word Space)

2. Computing similarities between them.

3. Employing the use of graph-based ranking algorithms [23] to remove the redundant sentences in the text.

The task involves simple mathematical computations, such as addition of vectors, and thus is far more effective than other algorithms based on semantic similarities, such as LSA based summarization that involves expensive matrix computations.

3.3.3 The Word Space Model

The Word-Space Model [24] is a spatial representation of word meaning. The complete vocabulary of any text (containing n words) can be represented in an n-dimensional space in which each word occupies a specific point in the space and has a vector associated with it defining its meaning.

The Word Space Model is based entirely on language data available. It does not rely on previously compiled lexicons or databases to represent the meaning. It only represents what is really there in the current universe of discourse. When meanings change, disappear or appear in the data at hand, the model changes accordingly. If an entirely different set of language data is used, a complete different model of meaning is obtained.

The Word Space Hypotheses The Word Space is so constructed that the following two hypotheses hold true.

1. *The Proximity Hypothesis:* The words which lie closer to each other in the word space have similar meanings while the words distant in the word space have dissimilar meanings.

2. *The Distributional Hypothesis:* Once the language data is obtained, the word space model uses the statistics about the distributional properties of the words. The words which are used within similar group of words (i.e. similar context) should be placed nearer to each other.

An environment in linguistics is called a context. A context of a word can easily be understood as the linguistic surrounding of the word. As an illustration, consider the following sentence.

'A friend in need is a friend indeed' If we define the context of a focus word as one preceding and one succeeding word, then the context of '*need*' is '*in*' and '*is*', whereas the context of '*a*' is '*is*' and '*friend*'.

To tabulate this context information a co-occurrence matrix of the following form can be created, in which the (i,j)th element denotes the number of times word i occurs in the context of word j within the text.

	Co-occurrents					
Word	a	friend	in	need	is	indeed
a	0	2	0	0	1	0
friend	2	0	1	0	0	1
in	0	1	0	1	0	0
need	0	0	1	0	1	0
is	1	0	0	1	0	0
indeed	0	1	0	0	0	0

Figure 3.3 Word Space Model

Here the context vector for 'need' is [0 0 1 0 1 0] and for 'a' is [0 2 0 0 1 0]. They effectively sum up the environments (contexts) of the words in question, and can be represented in a six-dimensional geometric space (since the text contains 6 words). A context vector thus obtained can be used to represent the distributional information of the word into geometrical space. (Note

that this is similar to assigning a unary index vector to '*is*' ([0 0 0 0 1 0]) and to '*in*' ([0 0 1 0 0 0]) and adding them up to get the context vector of '*need*'.)

Similarity in Mathematical Terms Context vectors as such do not convey any beneficial information. They just give the location of the word in the word space. To get a clue on 'how similar the words are in their meaning' a similarity measure of the context vectors is required.

Various schemes, such as scalar product of vectors, Euclidean distance, Minkowski metrics, can be used to compute similarity between vectors.

Cosine of two vectors is used to calculate similarity between vectors.

$$\mathrm{sim}_{\cos}(x,y) = \frac{x \cdot y}{|x||y|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}$$

The cosine measure corresponds to taking the scalar product of the vectors and then dividing by their norms. The cosine measure is the most frequently utilized similarity metric in word-space research. The advantage of using cosine metric over other metrics to calculate similarity is that it provides a fixed measure of similarity, which ranges from 1 (for identical vectors), to 0 (for orthogonal vectors) and -1 (for vectors pointing in the opposite directions). Moreover, it is also comparatively efficient to compute.

The sentences that are most similar to other sentences and to the whole document are considered to be most important and such sentences constitute the summary of the document.

Chapter 4

SEMANTIC SEARCH AND INDEXING

"The ultimate search engine would basically understand everything in the world, and it would always give you the right thing. And we're a long, long ways from that."

Larry Page (Cofounder Google)

4.1 Semantic Search

The Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. It is the idea of having data on the Web defined and linked in a way that it can be used for more effective discovery, automation, integration, and reuse across various applications. In particular, the Semantic Web will contain resources corresponding not just to media objects (such as Web pages, images, audio clips, etc.) as the current Web does, but also to objects such as people, places, organizations and events. Further, the Semantic Web will contain not just a single kind of relation (the hyperlink) between resources, but many different kinds of relations between the different types of resources mentioned above.

4.1.1 Main Features

Documents vs Real World Objects: The Semantic Web is not a Web of documents, but a Web of relations between resources denoting real world objects, i.e., objects such as people, places and events. In the first example we have objects such as the city of Paris, the musician Yo-Yo Ma, an auction event, the music album Appalachian Journey, etc. In the second example, we have the person Eric Miller, the W3C Semantic Web Activity, the organization W3C, the city of Dublin Ohio, etc.

Human vs Machine Readable Information: There are resources corresponding to Eric Miller. This is not the string"Eric Miller", but a resource denoting a person. There are many people with the name Eric Miller. This denotes only one particular person with that name. The salient point about the Semantic Web is that it contains rich machine readable information about these resources.

Relation between the HTML & Semantic Web: The Semantic Web is an extension of the current Web. There is a rich set of links from the nodes in the Semantic Web to HTML documents. These relations typically connect a concept in the Semantic Web with the pages that most pertain to it.

Distributed Extensibility: An important aspect of the Semantic Web is that different sites may contribute data about a particular resource. This *distributed extensibility* is a very important aspect of the Semantic Web.

Of course, this feature leads to problems of its own. In a world where anyone can publish anything, a lot of what gets published cannot be trusted. In the current Web, we, as humans, use our intelligence, invoking concepts of brand, who recommended what, etc. to decide whether to believe what a Web site says. Programs, on the other hand, being relatively unintelligent, do not have recourse to all these facilities to decide whether to believe the data from a new site on the Semantic Web.

4.1.2 The Types Of Searches

As with the WWW, the growth of the Semantic Web will be driven by applications that use it. Semantic search is an application of the Semantic Web to search. Search is both one of the most popular applications on the Web and an application with significant room for improvement. We believe that the addition of explicit semantics can improve search. Semantic Search attempts to augment and improve traditional search results (based on Information Retrieval technology) by using data from the Semantic Web.

Traditional Information Retrieval (IR) technology is based almost purely on the occurrence of words in documents. Search engines like Google, augment this in the context of the Web with information about the hyperlink structure of the Web. The availability of large amounts of structured, machine understandable information about a wide range of objects on the Semantic Web offers some opportunities for improving on traditional search.

Before getting into the details of how the Semantic Web can contribute to search, we need to distinguish between two very different kinds of searches[25].

Navigational Searches: In this class of searches, the user provides the search engine a phrase or combination of words which s/he expects to find in the documents. There is no straightforward, reasonable interpretation of these words as denoting a concept. In such cases, the user is using the search engine as a navigation tool to navigate to a particular intended document.

We are not interested in this class of searches.

Research Searches: In many other cases, the user provides the search engine with a phrase which is intended to denote an object about which the user is trying to gather/research information.

There is no particular document which the user knows about that s/he is trying to get to. Rather the user is trying to locate a number of documents which together will give him/her the information s/he is trying to find. This is the class of searches we are interested in.

Example: A search query like "W3C track 2pm Panel" does not denote any concept. The user is likely just trying to find the page containing all these words. On the other hand, search queries like "Eric Miller" or "Dublin Ohio", denote a person or a place. The user is likely doing an research search on the person or place denoted by the query.

Semantic search attempts to improve the results of research searches in 2 ways.

Traditional search results take the form of a list of documents/ Web pages. We augment this list of documents with relevant data pulled out from Semantic Web. The Semantic Web based results are independent of and augment the results obtained via traditional IR techniques. So, for example, a search for Yo-Yo Ma might get augmented with his current concert schedule, his music albums, his image, etc. The search phrase in Research Searches typically denotes one (or occasionally two) real-world concepts. We believe that it might be useful for the text retrieval part of the search engine to have an understanding of these concepts denoted by the search phrase. Understanding the denotation can help understand the context of the search, the activity the user is trying to perform, drive expectations on the categories of documents (pertaining to the object) likely to exist, etc.

Current statistics show that the number of Web pages is in excess of one billion (Madhavan et al., 2007). With such a huge amount of data available some means had to be put in place so that exact information, relating to a particular purpose, could be searched for and retrieved. This need for relevant information spawned the production of search engines dedicated to sift through the Web and pick out documents and content that satisfy the requirements given to the engine by a user. However, a successful search for information can only be as good as the ability of the search engine to retrieve that information. Web pages, and any form of information given as text, is only read by a machine but is not understood by a machine. This means that although a text search for a word will be able to find that particular word, the machine cannot attach any meaning to the words it finds or is looking for. This severely limits the ability of a machine to retrieve data according to the needs of a human.

For example, if a person wanted to search a document or group of documents for information about 'men's clothes' they would have to tailor their thinking to extract the words

that they thought an article that talked about this subject would contain, such as shirts, ties, coats, sizes, materials, price and so on. If, however, the machine somehow knew that a shirt was an item of clothing, or that coats come in four sizes and knew that ties could be of silk or cotton then the machine could automatically give all this information without the need for multiple searches.

4.1.3 Problems of Search and Retrieval on the Web

One of the major uses of the World Wide Web today is to use search engines to find out about information. Due to the increase in the number of Web pages available, finding the desired information can often be an arduous and complex task. There are a number of problems with the current approach to finding results on the web:

- ***The most basic problem is that the information that you are looking for cannot be found. This could happen for a number of reasons****:*

1. The information simply does not exist on the web.

2. The search engine could not find the information that does exist.

3. The information is in a web page that is not on the first page of returned results and the user does not try and look on through the next few results pages.

Passin gives an example where he could not find a replacement part for an old stove, even though there was a shop with a website that sold the part.

- ***There may be too many irrelevant results returned.***

The ratio of the number of relevant results retrieved to the total number of irrelevant and relevant results retrieved is called the *precision* of a search. As a brief example, if a person wanted to know about the Isle of Wight Mosque, then entering this search term into Google produces a total number of 38600 results. However, out of these there are only 2 results that are about the Mosque. This gives a precision value of 0.00005%. The other results are pages that contain the words 'Isle of Wight' and the word 'Mosque' with little or no connection between them.

- ***Only some of the relevant results to the query are returned.***

The ratio of the number of relevant results retrieved to the total number of relevant results indexed by the search engine is called the *recall* of a search. This value is harder to measure as it is virtually impossible to calculate how many relevant results to a search exist on the web. However, Clarke and Willett (1997) produce a relative estimation of recall based on pooling the results of a number of search engines to the same query. Shafi and Rather (2005) use this method to compare the precision and recall of five search engines in the retrieval of scholarly information in the field of

biotechnology.

- ***The search engine searches for the query term in the wrong context and returns results in that context.***

There are many cases where words have more than one meaning. If a user does not correctly add extra keywords to their search to contextualise the topic, they could receive anomalous results.

For example, if somebody wanted to know about the Greek deity 'Nike', simply entering this word into a search engine will return results about the sportswear company Nike.

At present, Google has 46.3% (Sullivan 2006) of the search engine market. The reason for Google's success is claimed to be down to the PageRank algorithm (Brin et al. 1998) which ranks pages, not just by keyword frequency, but also according to how many pages link to a site, how many hits a site has and how often a site is updated, as well as other varying criteria (Arnold 2005). However, statistics also show that other search engines return results that are not found by Google (Notess 2002). This all means that the end user cannot make best use of the Web since the information or knowledge that the user requires is either difficult to find or search engine dependent.

The lack of a suitable search engine for the web stems from the fact that all search engines rely heavily on keyword frequency, and in essence, string matching to find their results. When a person enters the term 'football' into a search engine it has no knowledge about the concept of football, it will base its results on the number of times this keyword is mentioned in a document. There are algorithms that enhance this approach somewhat but the underlying principle is the same (Langville & Meyer 2004).

What is needed is a system that knows that football is a type of sport and that it has a World Cup and that it is played by 11 players etc. This kind of knowledge awareness is exactly what The Semantic Web is trying to achieve. If there was ontology for football then all of the above facts could be easily modelled and used to enhance query results. Then, if someone was to make a query about football, the system would try and attain the concept of the search such as, is the person looking for football tickets? Are they looking for football kit? Or are they interested in a particular team? And so on.

The Semantic Web promises a new generation of World Wide Web infrastructure that will make it possible for machines to 'understand' the data on the web instead of merely presenting it. In order to encourage the increase of semantic web technologies there have been

suggestions that a 'killer app' may be needed to convince those that are still unsure about the benefits that semantic web technologies can bring.

The search engine is an example of a potential 'killer app' that has been responsible for increased usage of the current web. As described in this section there are a number of problems with searching the Web today. Despite the improving quality of modern search engines, statistics show that only 17% of people find exactly the information they were looking for (Fallows, 2005). Furthermore, a study has shown that the recall of some search engines can be as low as 18% (Shafi & Rather, 2005). There is therefore a need to improve the quality of search results and user experience. The Semantic Web provides an opportunity to achieve such a goal. The use of RDF and OWL as knowledge representation formats can provide structured content to describe a given domain or set of domains. Using this knowledge, it should be possible to add a sense of 'understanding' to a search engine when searching for results whose knowledge has been partly or fully described in a knowledge representation format.

In the past, such a proposal may not have been viable due to the lack of ontologies and RDF resources available on the web. However, at the present time there are estimated to be more than 5 million RDF or OWL documents available on the Web (Ding, 2006). Even if most of those documents contain knowledge about a limited set of concepts, RDF data from sources such as DBpedia and Wordnet provide a suitable base from which to begin exploring the enhancement that can be made to ordinary Web searches. More importantly, there will be a number of knowledge bases that will be used to store RDF instance data and OWL ontologies that have the ability of being

4.2 Indexing

When the users fire a query the system does not start searching the documents. Even in a simple search engine documents are indexed beforehand. Index is basically a table that links words to documents. These tables or data structures allow system to quickly find the list of documents that contain words that user is looking for. These are the data structures that speed up the searching. There are two types of index

1. Inverted Index

Words	List of Documents
Word1	Doc1, Doc3, Doc 3, Doc 6…
Word2	Doc2, Doc10 , Doc4, Doc11…

Figure 4.1 Inverted Index

2. Forward Index.

Documents	List of words
Doc1	Word1 Word5 Word9…
Doc2	Word2 Word13 Word14…

Figure 4.2 Forward Index

Inverted index contains list of documents for each word found the document collection or corpus. Forward Index on the other hand contains list of words contained in each of the documents. These indexes normally contain more information that just this. They may contain more information such as number documents, unique words in each document, number of documents in which a given word was found etc. Both indexes are useful for an efficient search engine. To enhance the performance of search many other techniques such as hierarchical

Fields	Doc1	Doc2
Filename	Doc1.txt	Doc2.txt
path	/doc1.txt	/doc2.txt
PERSON		gates,Sachin, bill
BUSINESSMAN		gates, bill
ELEC._DEVICE	gates, resistor	
SEMICON._DEVICE	gates	

Figure 4.3 Semantic Index

indexing, hashing etc. are used. Indexes often contain more than one field. Fields are created when all words in a document are not to be treated equal. For example, words found in the

description MetaTag of an HTML document may be given higher weight. When fields are included in the index a separate index is created for each field in case of inverse index. An in case of forward index the list also specifies the field in which the word was found.

Indexing for semantic documents is much more complex. The document is indexed not only for words but also for named entities, their classes and relations between named entities etc. All this semantic information must be encoded in the index such that search engine can quickly take advantage of it. In fact it does not matter how much semantic information is present in the semantic document if it is not indexed properly. If the indexer is a simple document indexer than the search results will be no different than a syntactic search engine. One way to index a semantic document is to create a separate field for each class in the ontology and to index all the named entities that are direct or indirect instances of a class in the field for that class. This type of index allows user to express the type of the entity user is searching to any degree of precision as permitted by the ontology.

Chapter 5

TOOL ARCHITECHTURE AND DESIGN APPROACH

An algorithm must be seen to be believed.

- Donald Knuth

5.1 Tool Architecture

This section describes the overall architecture of the tool[26]. The functional diagram[27] for the system is shown in Figure 5.1. It consists of several components. There are four data stores and five processes. The four data sources are Ontology, Document Collection and two indexes Document Index and Instance Index. The Five processes are Annotator, Evaluator, Indexer, Summarizer and finally the searcher. Except searcher all other processes are offline processes. There collective job is to transform a document collection into a pair of index table that can be later on searched by the searcher to present the user with the information relevant to the user query.

These processes operate in a sequence. First the annotator annotates the document that is it embeds the semantic i.e. meaning related information in the documents using the ontology. Once the documents are annotated these are analyzed by evaluator and categorized and stored in different clusters/groups. Now indexer prepares the index of these annotated documents taking the advantage of both annotated documents and the ontology. Indexer also indexes the summary of the documents; this summary is produced by the summarizer. Finally when the user fires a

query searcher searches the index and ontology extract as relevant information as possible for user.

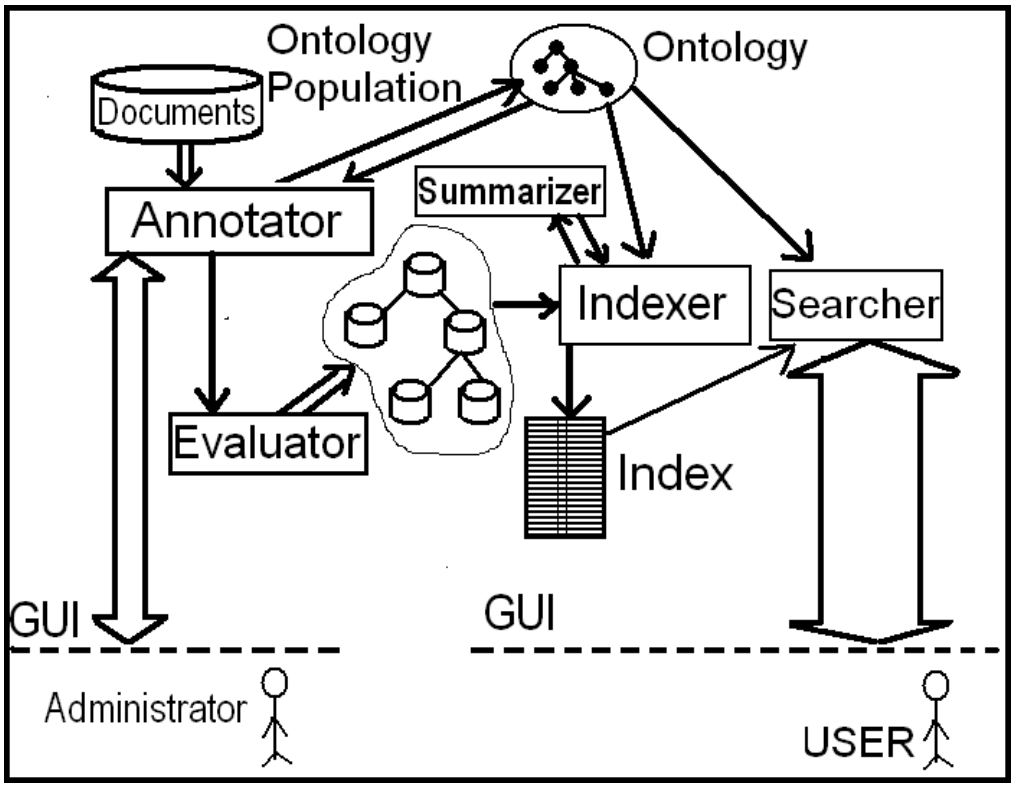


Figure 5.1 Tool Architecture

Role of Administrator: Administrator is person or a team of personnel employed by the organization. The Administrator interacts with the annotator, Indexer, Summarizer and Ontology Population. The Administrators job is to annotate the documents that are uploaded by the users on the web portal using Annotator. Once they are annotated the administrator runs the indexer which automatically prepares the index. The Administrator is also responsible for keep the ontology up-to-date and consistent. He runs the ontology population tool to add new instances to the ontology. Once the new instances are added then administrator has to manually update the relations present in the ontology.

Role of User: The User consists of the people who visit the web portal. User uploads the document on the portal. They also retrieve information from the portal using the search engine.

Here in this thesis I am going to explain with design approach & implementation details the following sections of the tool.

- **Indexing Annotated Documents**
- **Document repositioning**
- **Document Summarizing**
- **Semantic Searching**

Here we also present the block diagram with implementation of the tool.

As Described in the above section, the output of Semantic Annotator is a Document in which mark up has been added for each Named Entity identified in the document. This document contains semantic information. Before Indexing the document is first analyzed and stored at a specific location according to its content by Evaluator Process. It is then indexed by the indexer which calls the summarizer process when needed to create the summary of the document. It is then searched by the searcher. Therefore first the Evaluator is explained and then Indexer. After this summarizer is explained and at last the searcher.

5.2 Design Approach

5.2.1 Ontology based document repositioning (Evaluator)

Once the documents have been annotated it has been enriched by the information from the ontology. We can use this information from ontology to find out to which category the uploaded document belongs. We use following algorithm to do it.

Method 1 Evaluate Document

1. *Create a table with a row for each of the class in the ontology. We fill the first column with name of the class and second column with the number of annotations that belong to this class and all the subclasses of this class.*
2. *set cur class to topmost class*
3. *create a list of all direct subclasses*
4. *Find two winners first and second.*
5. *if number of annotations that belong to first is more than twice of the second*
    a. *Set cur class to first class and go to step 3.*

   Else

   Return with cur class as the type of the document.

The above algorithm ensures an important property of the classification.

That is if in the ontology class c1 is a super-class of class c2, and if document is of type c2 then it is also of type c1.

The working of the algorithm is shown in fig 5.2. In this algorithm the test criterion is applied at each level in the ontology until the check fails, at this point the current class becomes the type of the document. The criteria is that number of annotations that belong to current class should exceed twice the number of annotations that belong to any other class at that level.

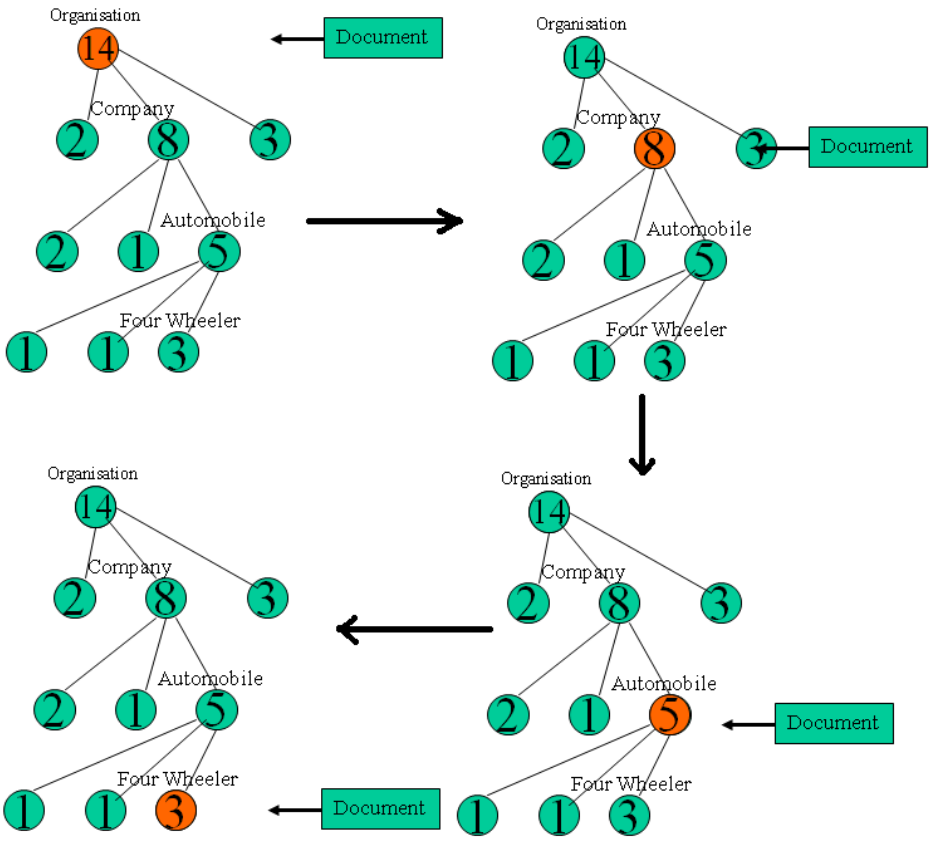


Figure 5.2 Document Classification Algorithm

Once the type of the document is known it is stored in the file system under the folder/directory of its type as name. The folders/directories have hierarchy that is same as the class hierarchy in the ontology. For this prototypical application we save the documents in a directory that corresponds to the type just found for this class.

5.2.2 Indexer

Indexer is a module that creates index which is used by the search engine to reply to user queries. It is the final step in the processing of documents. An index is a set of documents; each document is a set of fields. The structure of the index is shown in figure 5.3

Figure 5.3 Index Format

The first field is the name of the file. The second field is the Path, where the file is stored. The third field stores the time and date when the file was last modified. Fourth field stores the summary of the document which is calculated by the summarizer which is described in next section. There is a filed for each class in the ontology. These fields store the named entities in the document that are direct or indirect instances of respective class. These are the fields that empower search engine with semantic features.

A separate index for the instances in the ontology was also created. This was done to find out the class of a particular ontology quickly online without requirement for loading the whole ontology in the memory. We simply create two fields one with name of the instance in the ontology and the other with content that we should match in document for that Instance.

5.2.3 Summarizer

It would be very helpful to users if we give the summary of the documents that are listed by search engine instead of snippets. We use the algorithm presented in [2]. This paper presents the randomized approach to document summarization. Randomized approach does reduce the space requirements but at the cost of precision. We implement a mixture of non-randomized and randomized indexing,

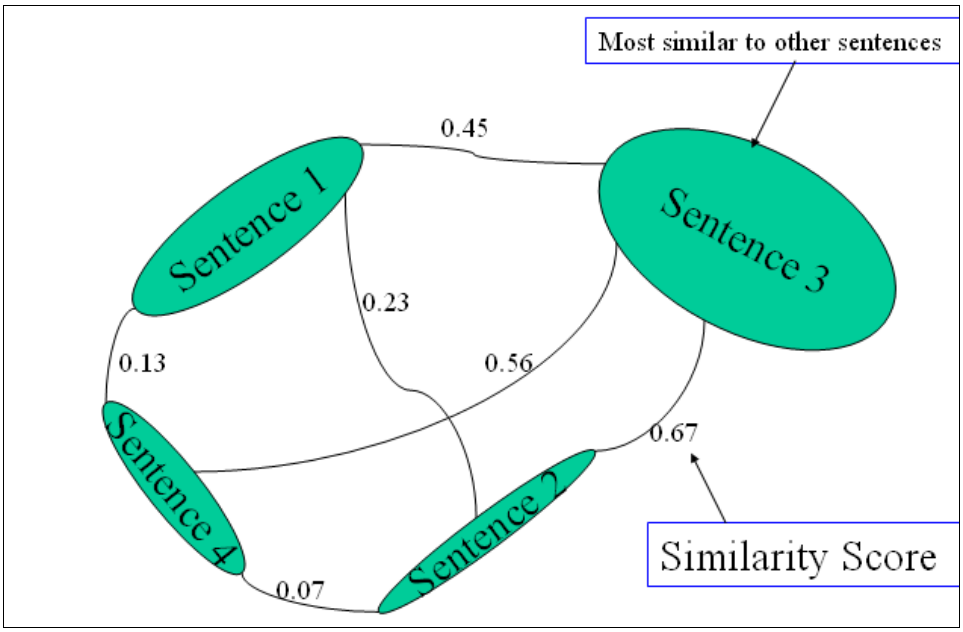


Figure 5.4 Summarize Algorithm

For summarization of documents, first similarity of each sentence with respect to each other sentence in the document is calculated. These similarity scores are later used as edge weights for the graph that is constructed. This graph has sentences as nodes. For initial node weights, importance of sentence in the document is calculated. Final scores are calculated by repeatedly applying weighted page rank which is a modified algorithm of the famous page rank algorithm [5].

The sentence with highest weight denoted the most important sentence i.e the sentence most similar to other sentences and import in the document. This sentence must be in the summary. Summary is constructed by sorting the sentences by their weights and taking top few sentences.

The algorithm used is :

1. ***Construct Completely Connected Graph of sentences***
2. ***Calculate Edge Weights***
    - *Construct Context vector for each word*
    - *Construct Semantic Vector For each Sentence (**sv**)*
    - *DotProduct(**sv_i,sv_j**)*
3. ***Calculate Node Weights***
    - *Sentence Context Vector (**scv**) , Average Document Vector (**adv**)*
    - *DotProduct(**acv,adv**)*
4. ***Apply Weighted Page Rank Algorithm iteratively**.*
    - *PWvi = (1-d) + d* Σ wij * PWvj /(Σwjk)*
5. ***Sort Results, Output Top few Sentences***

5.2.4 Semantic Searching

The Searcher is a module that is run every time user hits a query. It searches the index and the ontology to provide user with relevant information.

The main idea of semantic search is make information available in much easier way. Thus, reducing the time that user spends on the search engine. This is possible if the computer itself does a lot of processing on each document before it is presented to user. Documents are processed to analyze the document with respect to the user queries. So that only most relevant documents are presented to user.

To achieve above objective we add many features to our search engine. First it allows user to search a word in given context. Second, it gives summary of each document so that user can get idea of what is in the document without actually reading the whole document. Third, we categorize document themselves in addition to categorizing entities. Fourth, we provide search options that are semantically related to guide the user. These options allow users to expand their search thus making the search results more comprehensive.

Context Sensitive Search

In the current search engines when user fires a query the search engine searches for the word in the documents and prints them in order using some ranking method. Today's search engines fail when same word (text) means many different things. Like, if user searches for gates, she may mean electrical gates or Bill Gates. In traditional search engines user doesn't have power to specify this, but semantic search engine has.

Consider this scenario User fires a query "gates". And we have two documents indexed, one of them contains gates in the context of electronic gates and the other document contains gates in the context of Bill Gates. And suppose our ontology has two top level classes, ELECTRONIC_DEVICE and PERSON. PERSON has a sub class BUSINESSMAN and

Fields	**Doc1**	**Doc2**
Filename	Doc1.txt	Doc2.txt
path	/doc1.txt	/doc2.txt
PERSON		gates,Sachin, bill
BUSINESSMAN		gates, bill
ELEC._DEVICE	gates, resistor	
SEMICON._DEVICE	gates	

Figure 5.5 Context Sensitive Search

ELECTRONIC_DEVICE has a sub class SEMICONDUCTOR_DEVICE. And consider that index for these two documents have following structure.

Now user can specify any class to get more refined results. If she specifies PERSON only Doc2 will be presented .This reduces the number of documents returned by the searcher.

Summarizing the documents.

In order to further reduce the time user spends reading relevant documents, it's a good idea to provide summary of each of the document in the result set of the searcher. User can read the summary before deciding to open and read the entire document.

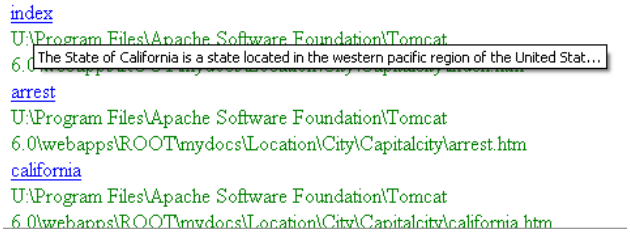


Figure 5.6 Summary

Summary is shown for the index document. These summaries were created offline during indexing, therefore it does not affect the query response time.

Document Classification

As described in section 5, documents are classified and stored in the file system under directories that form same hierarchy as the classes in the ontology. Thus a document that has been classified as of type 'City' is stored in a directory named 'CITY' which is a subdirectory of 'LOCATION'. Thus when the results of the user query are displayed, the path of the document

gives an indication of the content in the document, before the user even opens it. Therefore, if user can choose to open the document that appears to be closest to what he is looking for.

Semantically Related Search options

Above three features enable searcher to make sense of as much information user has about the information she wants, which it uses to streamline the results, thus reducing the number of documents returned. But in some cases user herself does not have enough information that is sufficient to guide search engine towards the documents that contain the information user is searching for. In such cases the searcher must provide the user some information. Consider following scenario.

User floats a query "bill", now this query is not sufficient for search engine give relevant information to the user. The word bill may mean almost anything; it may mean a electricity bill, telephone bill, Bill Gate or may be that user is interested in knowing about Microsoft or even about the America Corporate world. One solution is to accept some more information from user about the word "bill", this solution is presented by three features presented above in sections. But if user does not know anything more than this then searcher must provide some options. These options must be semantic/(meaning fully) related to the search query.

One way to generate semantically related options is to traverse ontology in an ordered way. We devise a measure of semantic closeness. It is measured by two variables VDist (Vertical Distance) and HDist (Horizontal Dist). Distance is number of edges in the path between two entities/instances in the ontology. When measuring vertical distance all edges are formed by IS_A or the taxonomic relations are considered and when measuring horizontal distance the edges are formed by non-taxonomic (all other relations except IS_A relations). Searcher component uses following algorithms to present user with semantically related options with a specified HDist and VDist by user.

Main Method

traverseOntology *(HDist, VDist, Instance)*

1. *create Empty String output*
2. *create Empty HashTable 'HHashTable'*
3. *output+=HTravel(HDist,Instance, HHashTable)*
4. *create Empty Hash Table 'VHashTable'*
5. *output+=VTravel(VDist,typeOf(instance),VHashTable)*
6. *return output;*

SubMethod 1

HTravel*(HDist, Instance, HHashTable)*

1. *create Empty String output*
2. *if(HDist>0)*
    a. *output+=Instance.getName()*
    b. *put Instance in HHashTable*
    c. *get All properties of Instance*
    d. *for each property get instance I*
    e. *output+=I.getName()*
    f. *if(instance is not in HHashTable)*
        i. *output+=HTravel(HDist-1,I,HHashTable)*
3. *return output*

SubMethod2

VTravel*(VDist, class, VHashTable)*

1. *create Empty String output*
2. *if VDist>0*
    a. *getAll the instances I*
    b. *for each I*
        i. *if(instance is not in VHashTable)*
            1. *output+=I.getName()*
    c. *put I in VHashTable*
    d. *output+=VTravel(VDist- 1,superclass(class),VHashTable)*

Chapter 6

IMPLEMENTATION DETAILS

Measuring programming progress by lines of code is like measuring aircraft building progress by weight.

- **Bill Gates (Cofounder Microsoft Co.)**

There are two main modules in the system. First is OntologyManager. It provides all the functions related to ontology. All other modules call functions of this module when ever they require any information related to ontology. The second module is AnnieManager/Named Entity Extractor. Its an interface to GATE's ANNIE plugin. It provides functions that are used to find annotations from the documents. The figure given shows the different modules of the tool.

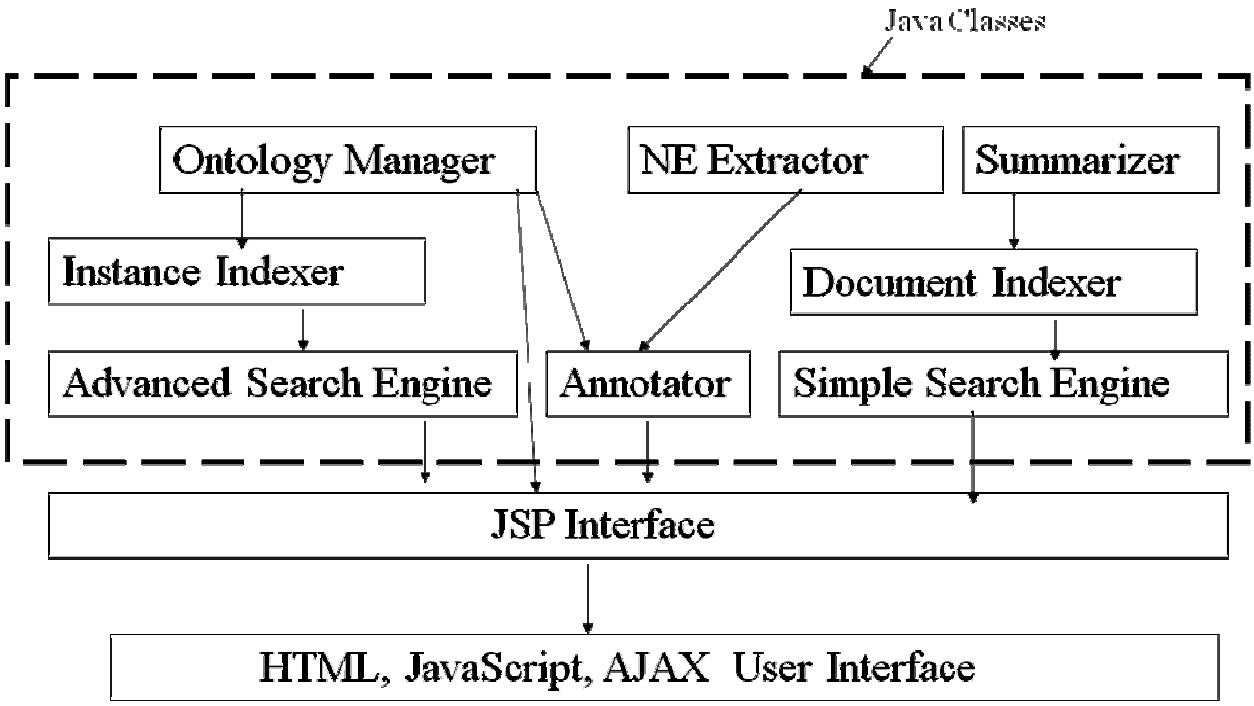


Figure 6.1 Modular View of Tool

6.1 OntologyManager

It has been implemented as a single OntologyManager.java class. It provides all the functions related to ontology. All other modules call functions of this module when ever they require any information related to ontology. It uses protégé OWL API to read/save ontology from the file. It also initializes many data-structures which are used by other modules.

1. Subclass Hash table.
   Structure

Class Name	Reference to list of subclasses

2. Instance Hash Table.

Instance Name	Reference to Instance Object

3. Class Hash Table

Instance Name	Reference of the Class

6.2 Named Entity Extractor

It is wrapper around GATE's ANNIE plug-in. It does the task of Initializing GATE, and loading ANNIE. It also provides functions to set the file to be annotated and to find annotations of that file. It provides a simple Iterator type interface to access Named entities of the set document. It has been implemented as *AnnieManager.java.* The named entities provided by this class are used by Annotator to create annotations in the document

6.3 Instance Indexer

It gets all the instances in the ontology from Ontology Manager. Creates a Index of them using Lucene. It saves the index in OntoIndex folder under Tomcat folder.

6.4 Document Indexer

It has been implemented as a multithreaded java class. It reads the documents from uploadedDocuments folder and adds each document one by one to the index. It also creates a queue of the names of document being indexed which is read by the GUI for displaying.

6.5 Summarizer

Summarizer has been implemented as a java class. It's an implementation of Extraction based summarization proposed in [22]. It is called by Document Indexer class for each document it indexes**.**

6.6 Evaluator

Evaluator has been implemented as a function in myAnnotate class. It takes the annotated document and finds out the class of that document. This class is passed to another function which finds out the path at which this document should be store.

6.7 Searcher

Searcher has been implemented as two different classes. One class performs simple search facilities provided by Lucene. Second class performs semantic search by using class fields of Document Index, Instance Index and Ontology.

6.8 Graphical User Interface

This tool has been developed as a web application. Application such as this requires a very rich Graphic User Interface. Its GUI has been made using JavaScript. It has different tabs for each of the module. Its GUI consists of tabs each from simple search, advance search, Indexing, Annotation and Ontology population. Below are the screen shots of the various tabs.

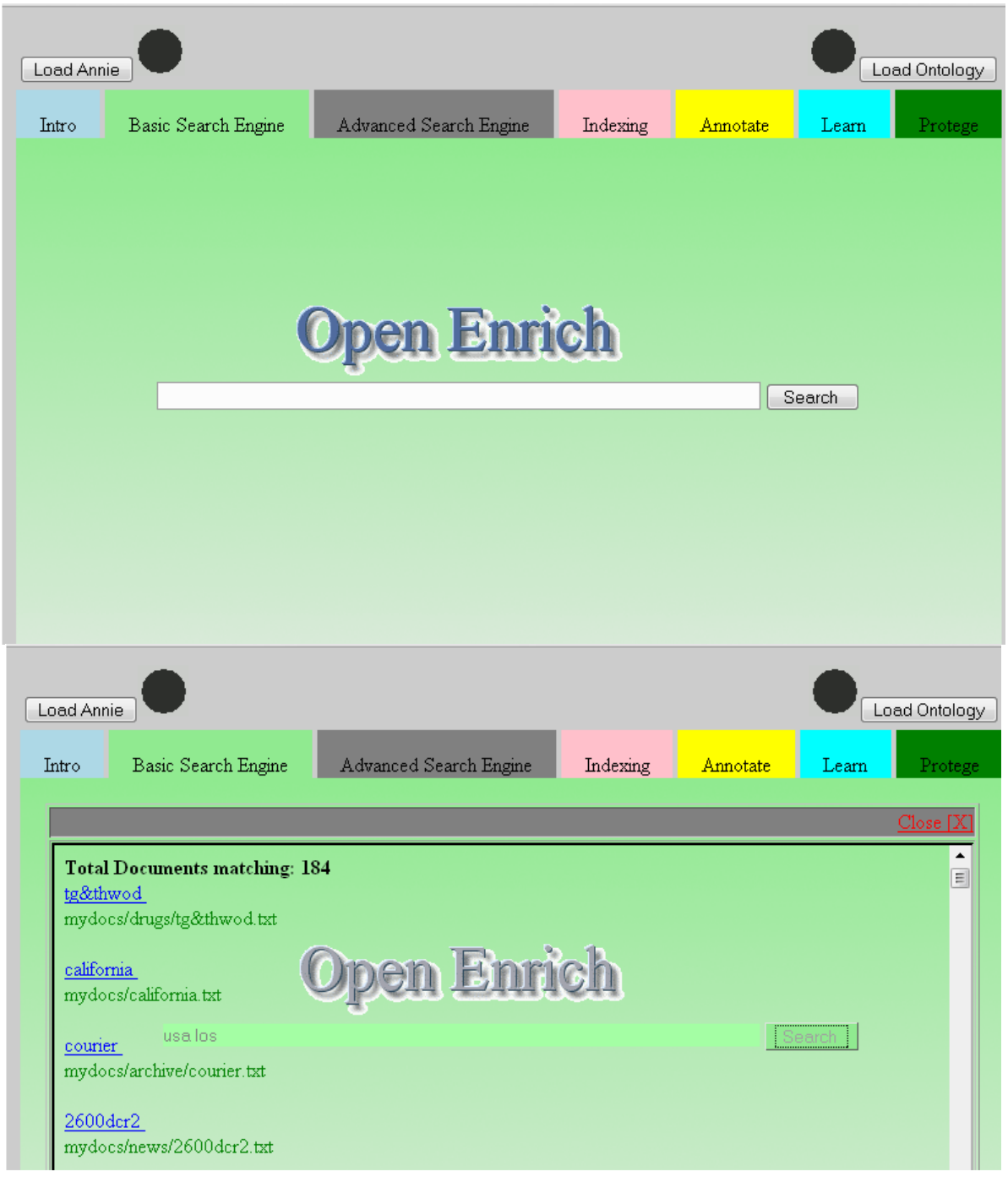
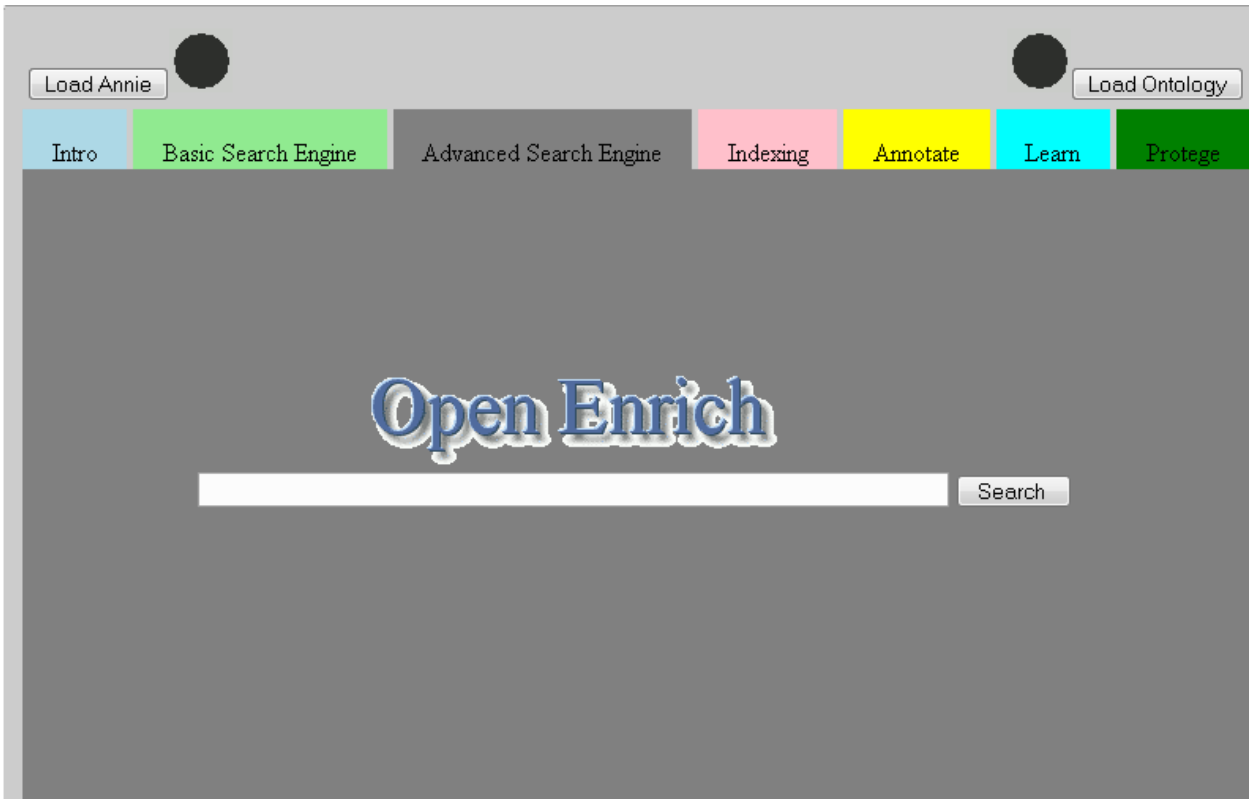


Figure 6.2 Simple Search Tab

Advance Search Tab



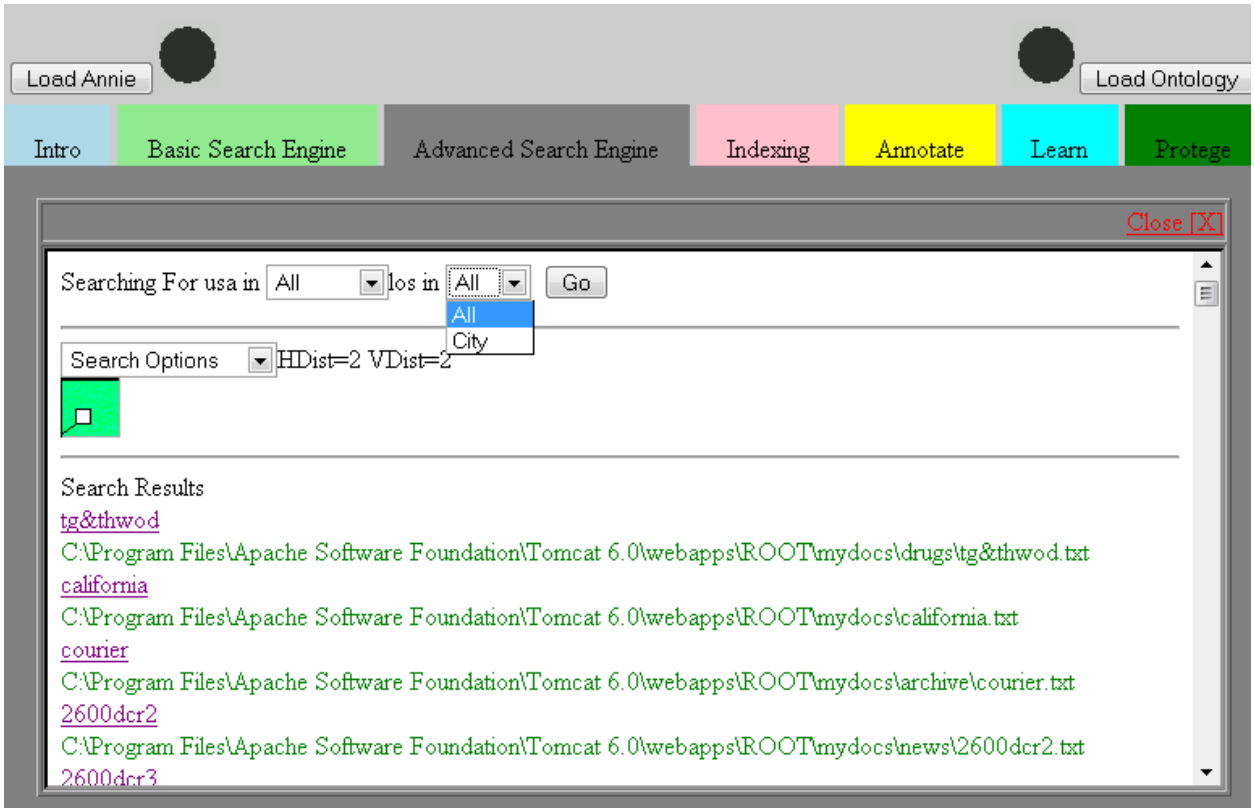
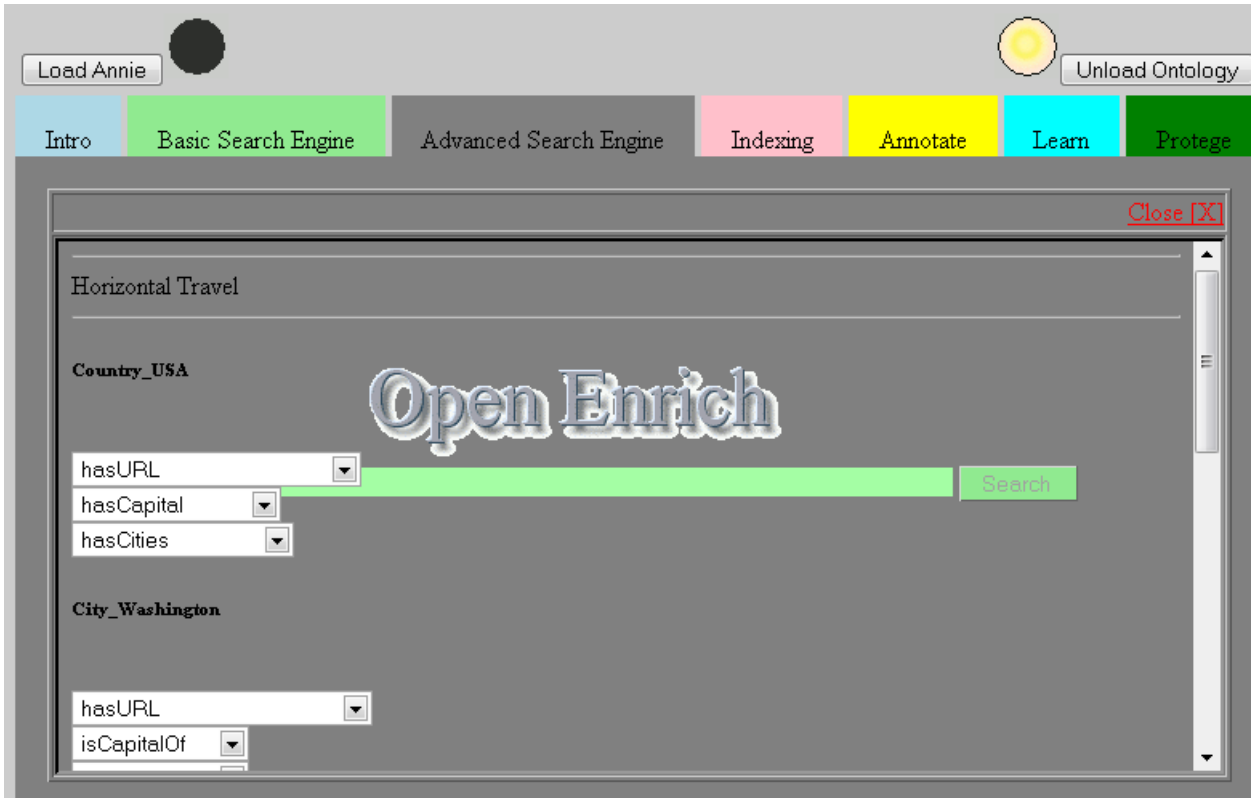
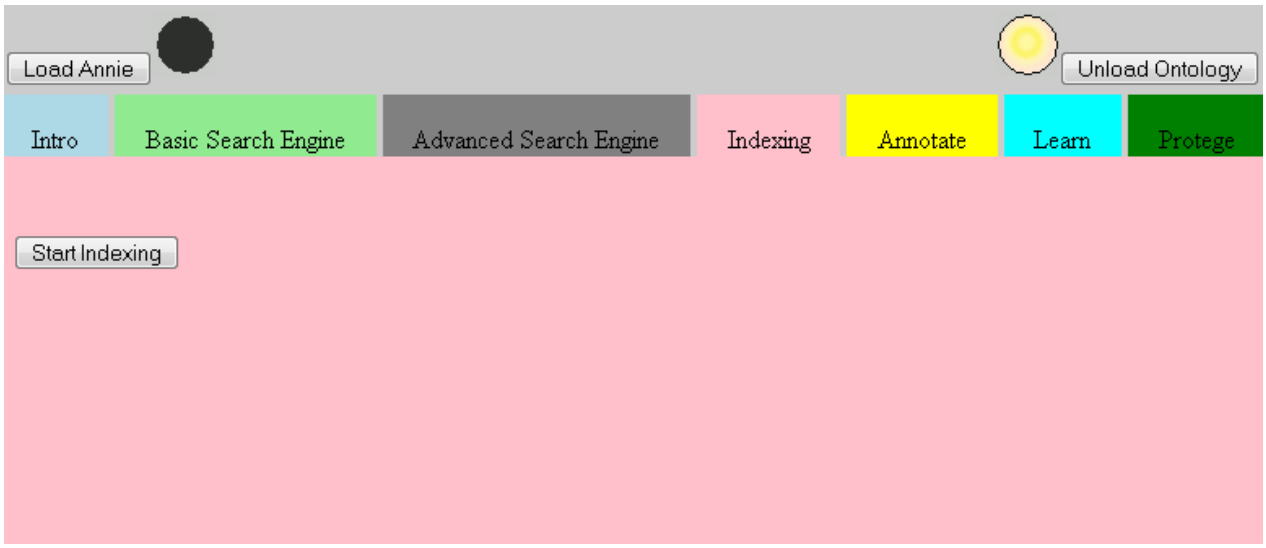


Figure 6.3 Advance Search Tab


Load Annie
Unload Ontology
Intro
Basic Search Engine
Advanced Search Engine
Indexing
Annotate
Learn
Protege
Close [X]
Horizontal Travel
Country_USA
Open Enrich
hasURL
Search
hasCapital
hasCities
City_Washington
hasURL
isCapitalOf


Load Annie
Unload Ontology
Intro
Basic Search Engine
Advanced Search Engine
Indexing
Annotate
Learn
Protege
Start Indexing

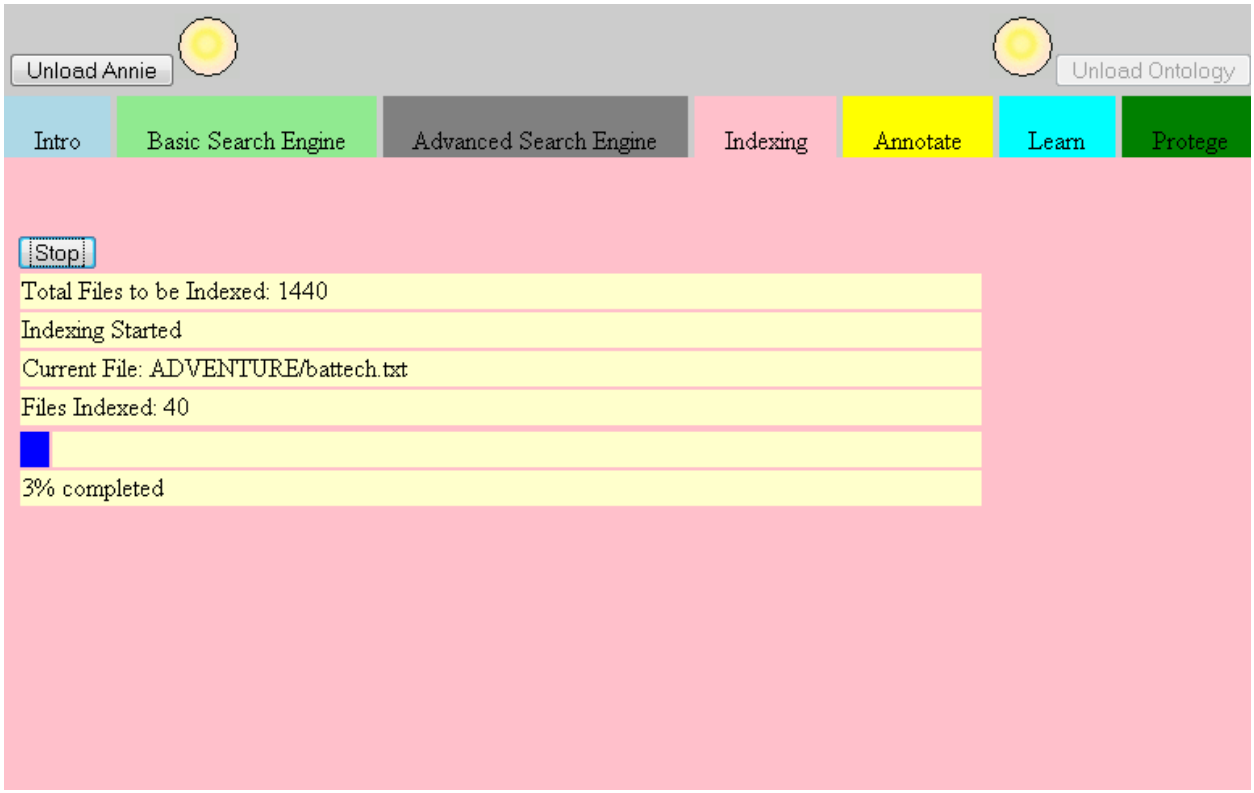


Figure 6.4 Indexing Tab

Chapter 7

CONCLUSION AND FUTURE WORK

"When you have a great and difficult task, something perhaps almost impossible, if you only work a little at a time, every day a little, suddenly the work will finish itself."

- Isak Dinesen (Author)

All above algorithms and tool have been implemented in java as the National Panchayat Portal is also developed using java. We use a combination of java [28], jsp and AJAX to create this tool. It provides a very good graphic user interface with all sorts of pop-up menus, trees and custom controls etc. We also integrated web protégé [29] with the application for free form editing of ontology.

Our approach makes use ontology to represent concepts and we use available tools and techniques of semantic web to improve information retrieval processes. The tool we have developed provides integrated framework for semantic annotation, semi automatic ontology population, which can further be extended to complete semantic web infrastructure including semantic search.

This work was appreciated a lot by NIC people. This tool will also be used as a prototype for the plug-in for the National Panchayat Portal to make it as India's first Semantic Portal[30].

We also received a certificate from them. We are really thankful for the help and support by NIC staff. Many issues regarding scalability, securities etc. are yet to be addressed. Below is a list for possible future extensions for this project.

Future Work

1. Co-reference Resolution
2. Relational to RDF conversion
3. Use of Attributes in semantic annotation
4. Ontology Evolution
5. Natural Language Processing
6. Using Different Ontologies on Same Domain.

PUBLICATIONS FROM THESIS

During the period of working over this project we interacted with International community working on semantic web. We discussed our approach for developing semantic web infrastructure with them and collected the reviews and worked over the suggestion send to us. Two research papers have been accepted in International conferences for presentation and will be published in their proceedings. Also we have communicated a journal paper, so that our work can be recognized and validated .These papers presents the methods and algorithm we have developed with detailed tool architecture that we have developed during the course of the project. The details of publications are as follows:

1. Conference Name **: "Semantic E-business and enterprise computing" SEEC 2008**

URL **: http://www.seec.eu/about.html**

Paper Title **: "An Approach to Semantic Web"**

Authors **: Nitin Nizhawan, Anuj Kumar, Daya Gupta.**

Location **: Kerala, India.**

Conference Date **: September 17-19, 2008**

2. Conference Name **: "Semantic Web & Web Services" SWWS 2008**

URL **: http://www.world-academy-of-science.org/worldcomp08/**

Paper Title **: "Semantic Search and Annotation Tool"**

Authors **: Nitin Nizhawan, Anuj Kumar, Daya Gupta.**

Location **: Las Vegas, USA.**

Conference Date **: July 14-17, 2008**

3. Journal Name : **International Journal on Knowledge Management**

URL : http://www.igi-pub.com/journals/details.asp?ID=4288

Paper Title **:"Semantic Web: An Approach"**

Authors : Nitin Nizhawan, Anuj Kumar, Kartar Jat, Daya Gupta

Status :Communicated

REFERENCES

[1] J. G. Carbonell and J. Siekmann, "*Intelligent information integration for the semantic web*", Springer, 2005

[2] W3C website , http://www.w3c.org

[3] Tim Berners-Lee. "*Realising the Full Potential of the Web*." W3C Document, URL http://www.w3.org/1998/02/Potential.html, Dec 1997.

[4] Tim Berners-lee. *Semantic Web Road Map*. W3C Design Issues. URL http://www.w3.org/DesignIssues/Semantic.html, Oct. 1998.

[5] A. Banerji, C. Bartolini, D. Beringer, V. Chopella, K.Govindarajan, A. Karp, H. Kuno, M. Lemon, G. Pogossiants, S. Sharma, S. Williams: *Web Services Conversation Language* (WSCL), HP, 2001.

[6] Tim Berners-Lee, James Hendler, and Ora Lassila. "*The semantic web*." Scientific American, 2001(5), 2001.

[7] Ramon F. Brena, Eduardo H. Ramirez, "*A Soft Semantic Web*", IEEE 2006.
Rada F. Mihalcea, Silvana I. Mihalcea, "Word Semantics for Information Retrieval: Moving One Step Closer to the Semantic Web".

[8] "WordNet", http://wordnet.princeton.edu/

[9] Iraklis Varlamis, Michalis Vazirgiannis, Maria Halkidi, Memeber, IEEE Computer Society, Benjamin Nguyen, "*THESUS, a Closer View on Web Content Management Enhanced with Link Smenatics"*, IEEE Transactions on Knowledge and Data Engineering, Vol. 16, No.6, June 2004

[10] D. Brickley and R.V. Guha, Resource Description Framework (RDF)Schema Specification 1.0, W3C Candidate Recommendation, March 27, 2000.

[11] Staab, S., Studer, R. editors: "*Handbook on Ontologies*", Springer, 2004.

[12] "Protégé", http://protege.stanford.edu.

[13] "*GATE, General Architecute for Text Engineering*", http://www.gate.ac.uk

[14] "*Lucene*", http://lucene.apache.org/

[15] Trent Apted , Judy Kay," *Automatic Construction of Learning Ontologies*" , Proceedings of the International Conference on Computers in Education (ICCE'02) 2002 IEEE.

[16] Xu Binfeng, Luo Xiaogang, Peng Chenglin and Huang Qian, "*Based on Ontology: Construction and application of Medical Knowledge Base*", 2007 IEEE/ICME Inrenational conference on complex medical engineering.

[17] "WordNet", http://wordnet.princeton.edu/

[18] Brooke Abrahams. Wei Dai," *Architecture for Automated Annotation and Ontology Based Querying of Semantic Web Resources*", Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05).

[19] Atanas Kiryakov, Borislav Popov, Damyan Ognyanoff, Dimitar Manov, Angel Kirilov, Miroslav Goranov*," Semantic Annotation, Indexing, and Retrieval*", Ontotext Lab, Sirma AIEOOD, 138 Tsarigradsko Shose, Sofia 1784, Bulgaria.

[20] Gobinda G. Chowdhury "*Natural Language Processing*",Dept. of Computer and Information Sciences University of Strathclyde, Glasgow G1 1XH, UK .

[21] Niladri Chatterjee, Shiwali Mohan, "*Extraction-Based Single-Document Summarization Using Random Indexing*", 19th IEEE International Conference on Tools with Artificial Intelligence, IEEE 2007,

[22] S. Brin and L. Page, "*The anatomy of a large-scale hypertextual Web search engine*", Computer Networks and ISDN Systems, 1998.

[23] Magnus Sahlgren, "*The Word-Space Model, Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces"*,2006

[24] R. Guha,Rob McCool,Eric Miller, *Semantic Search*,2002

[25] Kumar A. Nizhawan N. Gupta D."*Semantic Search and AnnotationTool*(SSAT)" .SWWS'08.

[26] Kumar A. Nizhawan N. Gupta D."*An approach to Semantic web*", SEEC 2008.

[27] Jayson Falkner, Kevin Jones, "*Servlets and Java ServerPages*", Addison-Wesley, 2000

[28] "*Protégé Web Interface*", http://protege.cim3.net/cgi-bin/wiki.pl?ProtegeWebBrowser.

[29] Matt Perry, Eric Stiles, "*SEMPL: A SEMantic PortaL*" 2004

APPENDIX A

AN INTRODUCTION TO

GATE

(General Architecture for Text Engineering)

URL: http://gate.ac.uk/

GATE is an infrastructure for developing and deploying software components that process human language. GATE helps scientists and developers in three ways:

1. by specifying an architecture, or organizational structure, for language processing software;

2. By providing a framework or class library that implements the architecture and can be used to embed language processing capabilities in diverse applications;

3. By providing a development environment built on top of the framework made up of convenient graphical tools for developing components.

GATE can be thought of as a Software Architecture for Language Engineering

Language Engineering (LE) may be defined as:

. . . the discipline or act of engineering software systems that perform tasks involving processing human language. Both the construction process and its outputs are measurable and predictable. The literature of the field relates to both application of relevant scientific results and a body of practice.

GATE as an architecture suggests that the elements of software systems that process natural language can usefully be broken down into various types of component, known as resources. Components are reusable software chunks with well-defined interfaces, and are a popular architectural form, used in Sun's Java Beans and Microsoft's .Net, for example. GATE components are specialised types of Java Bean, and come in three flavours:

• LanguageResources (LRs) represent entities such as lexicons, corpora or ontologies.

• ProcessingResources (PRs) represent entities that are primarily algorit- -hmic, such as parsers, generators or ngram modelers.

• VisualResources (VRs) represent visualisation and editing components that participate in GUIs.

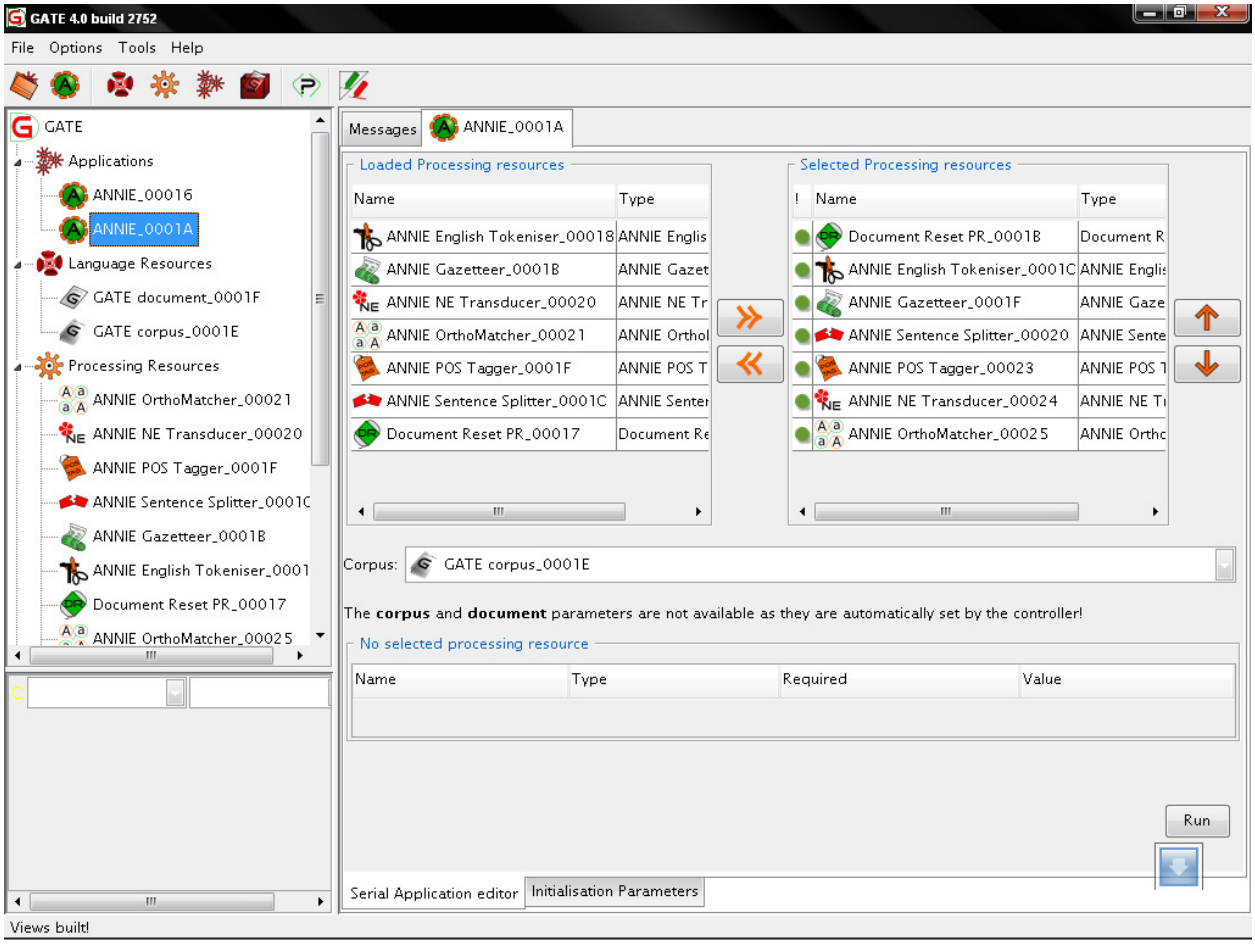


Figure A.1 GATE with ANNIE loaded

Built-in Components:

ANNIC

ANNotations In Context: a full-featured annotation indexing and retrieval system designed to support corpus querying and JAPE rule authoring. It is provided as part of an extention of the Serial Datastores, called Searchable Serial Datastore.

Machine learning API

A brand new machine learning layer specifically targetted at NLP tasks including text classification, chunk learning (e.g. for named entity recognition) and relation learning.

Ontology API

A new ontology API, based on OWL In Memory (OWLIM), which o_ers a better API, revised ontology event model and an improved ontology editor to name but few.

OCAT

Ontology-based Corpus Annotation Tool to help annotators to manually annotate documents using ontologies.

Alignment Tools

A new set of components (e.g. CompoundDocument, AlignmentEditor etc.) that help in building alignment tools and in carrying out cross-document processing.

New HTML Parser

A new HTML document format parser, based on Andy Clark's NekoHTML. This parser is much better than the old one at handling modern HTML and XHTML constructs, JavaScript blocks, etc., though the old parser is still available for existing applications that depend on its behaviour.

Ontotext Japec Compiler

Japec is a compiler for JAPE grammars developed by Ontotext Lab. It has some limitations compared to the standard JAPE transducer implementation, but can run JAPE grammars up to five times as fast.

Ontogazetteer

The Ontogazetteer, or Hierarchical Gazetteer, is an interface which makes ontologies "visible" in GATE, supporting basic methods for hierarchy management and traversal. In GATE, an ontology is represented at the same level as a document, and has nodes called classes.

Ontogazetteer Editor

This is for editing the class hierarchy of an ontology. it provides storing to and loading from RDF/RDFS, and provides load/edit/store of the class hierarchy of an ontology.

JAPE

Java Annotation Patterns Engine. JAPE provides finite state transduction over annotations based on regular expressions. JAPE is a version of CPSL

– Common Pattern Specification Language.

ANNIE: a Nearly-New Information Extraction System

GATE was originally developed in the context of Information Extraction (IE) R&D, and IE systems in many languages and shapes and sizes have been created using GATE with the IE components that have been distributed with it.

ANNIE components form a pipeline which appears in figure.

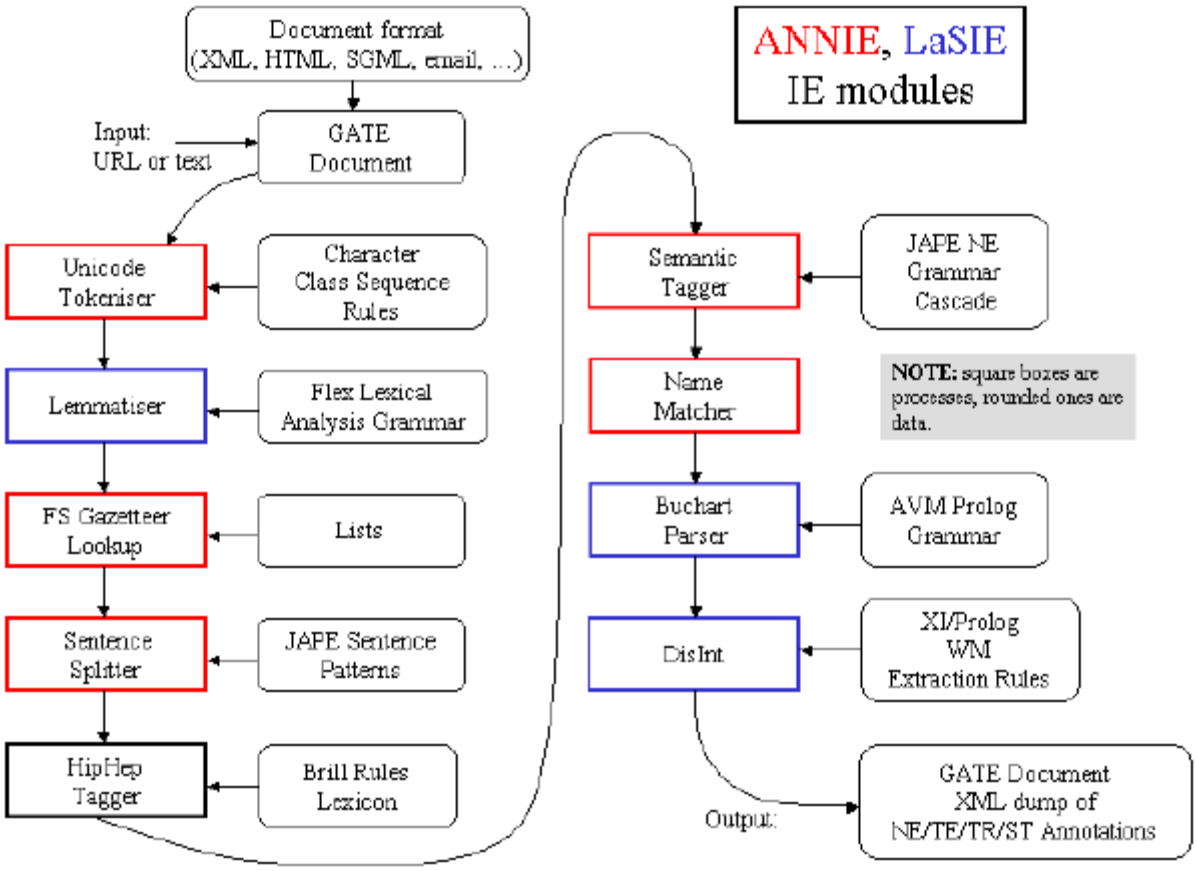


Figure A.2 Annie data flow diagram

Tokeniser

The tokeniser splits the text into very simple tokens such as numbers, punctuation and words of different types. For example, we distinguish between words in uppercase and lowercase, and between certain types of punctuation.

Gazetteer

The gazetteer lists used are plain text files, with one entry per line. Each list represents a set of names, such as names of cities, organisations, days of the week, etc. An index file (lists.def) is used to access these lists.

GATE includes resources for common LE data structures and algorithms, including documents, corpora and various annotation types, a set of language analysis components for Information Extraction and a range of data visualisation and editing components. GATE supports documents in a variety of formats including XML, RTF, email, HTML, SGML and plain text. In all cases the format is analysed and converted into a single unified model of annotation.

APENDIX B

An Introduction to Protégé

URL: http://protege.stanford.edu.//

Protégé is the latest tool in an established line of tools developed at Stanford University for knowledge acquisition. Protégé has thousands of users all over the world who use the system for projects ranging from modeling cancer-protocol guidelines to modeling nuclear-power stations. Protégé is freely available for download under the Mozilla open-source license.

Protégé provides a graphical and interactive ontology-design and knowledge-base–development environment. It helps knowledge engineers and domain experts to perform knowledge-management tasks. Ontology developers can access relevant information quickly whenever they need it, and can use direct manipulation to navigate and manage an ontology. Tree controls allow quick and simple navigation through a class hierarchy. Protégé uses forms as the interface for filling in slot values. The **knowledge model** of Protégé-2000 includes support for classes and the class hierarchy with multiple inheritance; template and own slots; specification of pre-defined and arbitrary facets for slots, which include allowed values, cardinality restrictions, default values, and inverse slots, metaclasses and metaclass hierarchy. In addition to highly usable interface, two other important features distinguish Protégé from most ontology-editing environments: its **scalability** and **extensibility**.

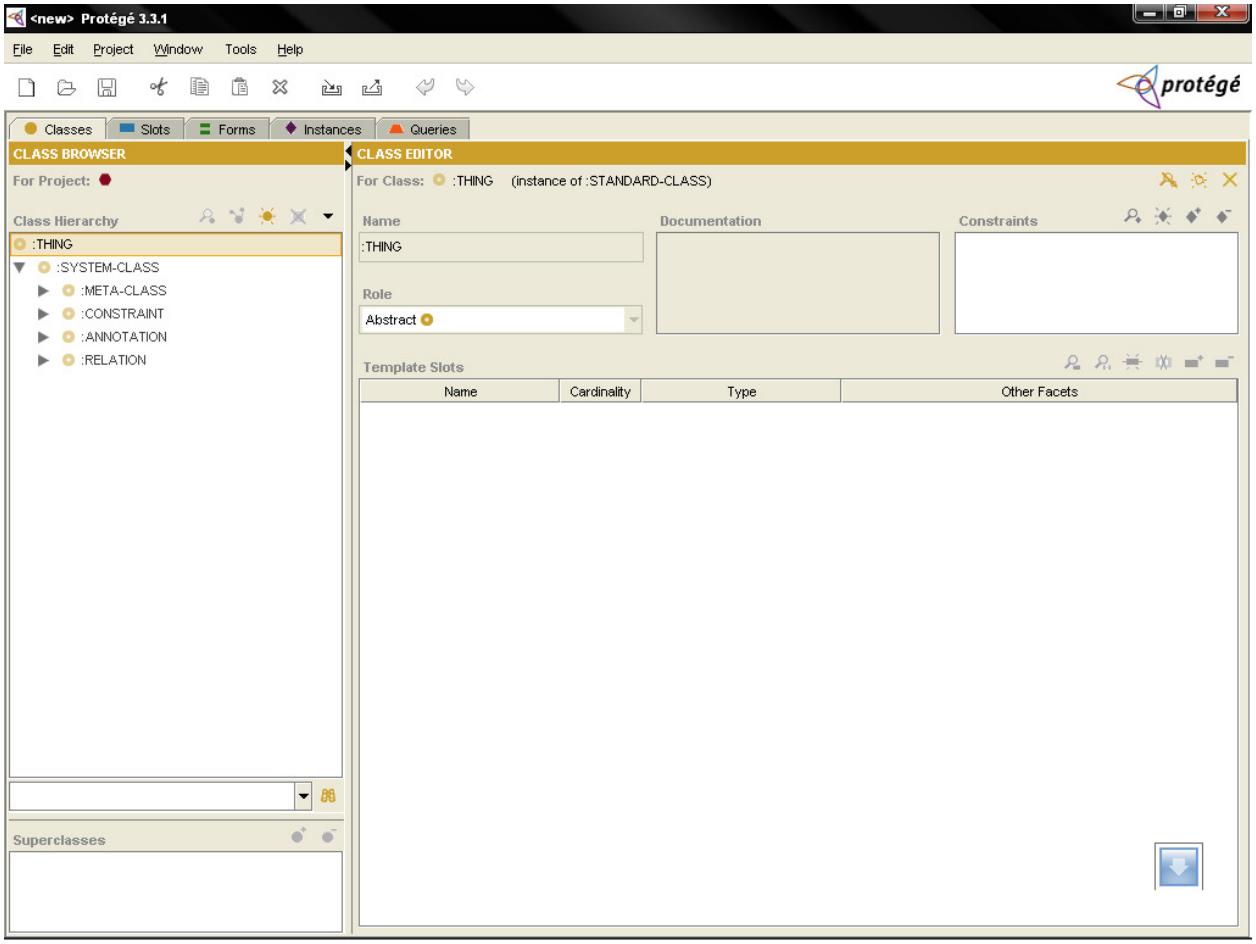


Figure B.1 Protege

One of the major advantages of the Protégé architecture is that the system is constructed in an open, modular fashion. Its component-based architecture enables system builders to add new functionality by creating appropriate plugins. The Protégé Plugin Library3 contains contributions from developers all over the world.

The Protégé-OWL API is an open-source Java library for the Web Ontology Language (OWL) and RDF(S). The API provides classes and methods to load and save OWL files, to query and manipulate OWL data models, and to perform reasoning based on Description Logic engines. Furthermore, the API is optimized for the implementation of graphical user interfaces. The Protégé-OWL API is centred around a collection of Java interfaces from the model package. These interfaces provide access to the OWL model and its elements like classes, properties, and individuals.

The API is designed to be used in two contexts:

- For the development of components that are executed inside of the Protégé-OWL editor's user interface
- For the development of stand-alone applications (e.g., Swing applications, Servlets, or Eclipse plug-ins)

Protégé is a flexible, configurable platform for the development of arbitrary model-driven applications and components. Protégé has an open architecture that allows programmers to integrate plug-ins, which can appear as separate tabs, specific user interface components (widgets), or perform any other task on the current model. The Protégé-OWL editor provides many editing and browsing facilities for OWL models, and therefore can serve as an attractive starting point for rapid application development. Developers can initially wrap their components into a Protégé tab widget and later extract them to distribute them as part of a stand-alone application.

APPENDIX C

AN INTRODUCTION TO LUCENE

Apache Lucene is a high-performance, full-featured text search engine library written entirely in Java. It is a technology suitable for nearly any application that requires full-text search, especially cross-platform. Lucene is a free/open source information retrieval library, originally implemented in Java by Doug Cutting. It is supported by the Apache Software Foundation and is released under the Apache Software License. Lucene has been ported to programming languages including Delphi, Perl, C#, C++, Python, Ruby and PHP.

While suitable for any application which requires full text indexing and searching capability, Lucene has been widely recognized for its utility in the implementation of Internet search engines and local, single-site searching. Lucene itself is just an indexing and search library and does not contain crawling and HTML parsing functionality. The Apache project Nutch is based on Lucene and provides this functionality; the Apache project Solr is a fully-featured search server based on Lucene.

At the core of Lucene's logical architecture is a notion of a document containing fields of text. This flexibility allows Lucene's API to be agnostic of file format. Text from PDFs, HTML, Microsoft Word documents, as well as many others can all be indexed so long as their textual information can be extracted

INDEX

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y