Advanced Intranet Search Engine

Nitesh Narayan

Supervisor:
Sasikumar Punnekkat

August 27, 2009
Abstract

Information retrieval has been a pervasive part of human society since its existence. With the advent of internet and World wide Web it became an extensive area of research and major focus, which lead to development of various search engines to locate the desired information, mostly for globally connected computer networks viz. internet. But there is another major part of computer network viz. intranet, which has not seen much of advancement in information retrieval approaches, in spite of being a major source of information within a large number of organizations.

Most common technique for intranet based search engines is still mere database-centric. Thus practically intranets are unable to avail the benefits of sophisticated techniques that have been developed for internet based search engines without exposing the data to commercial search engines.

In this Master level thesis we propose a "state of the art architecture" for an advanced search engine for intranet which is capable of dealing with continuously growing size of intranets knowledge base. This search engine employs lexical processing of documents, where documents are indexed and searched based on standalone terms or keywords, along with the semantic processing of the documents where the context of the words and the relationship among them is given more importance.

Combining lexical and semantic processing of the documents give an effective approach to handle navigational queries along with research queries, opposite to the modern search engines which either uses lexical processing or semantic processing (or one as the major) of the documents. We give equal importance to both the approaches in our design, considering best of the both world.

This work also takes into account various widely acclaimed concepts like inference rules, ontologies and active feedback from the user community to continuously enhance and improve the quality of search results along with the possibility to infer and deduce new knowledge from the existing one, while preparing for the advent of semantic web.
# Contents

**List of Figures**  
**Acknowledgement**  

## 1 Introduction
1.1 Goal of the work  
1.2 Outline of the thesis  

## 2 Search Engines
2.1 Architecture and working principle of basic Information Retrieval (IR) system  
2.2 Challenges  
2.3 Ongoing Research and future  

## 3 Artificial Intelligence (AI)
3.1 Software Agents  
3.2 Logical Inference  
3.3 Logic  
3.3.1 Propositional logic  
3.3.2 First-Order logic  
3.4 Ontology  

## 4 Natural Language Processing
4.1 Speech Segmentation  
4.2 Part-of-speech Tagging  
4.3 Word sense disambiguation  
4.4 Parsing  
4.4.1 Probabilistic Parser  
4.4.2 Logic Based parsing  

## 5 Open Source Tools
5.1 Stanford parser  
5.2 Wordnet lexicon  
5.3 Swi-Prolog  

---

### List of Figures

<table>
<thead>
<tr>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>iii</td>
</tr>
</tbody>
</table>

### Acknowledgement

<table>
<thead>
<tr>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>iv</td>
</tr>
</tbody>
</table>

### 1 Introduction

<table>
<thead>
<tr>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Goal of the work</td>
</tr>
<tr>
<td>1.2 Outline of the thesis</td>
</tr>
</tbody>
</table>

### 2 Search Engines

<table>
<thead>
<tr>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Architecture and working principle of basic Information Retrieval (IR) system</td>
</tr>
<tr>
<td>2.2 Challenges</td>
</tr>
<tr>
<td>2.3 Ongoing Research and future</td>
</tr>
</tbody>
</table>

### 3 Artificial Intelligence (AI)

<table>
<thead>
<tr>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Software Agents</td>
</tr>
<tr>
<td>3.2 Logical Inference</td>
</tr>
<tr>
<td>3.3 Logic</td>
</tr>
<tr>
<td>3.3.1 Propositional logic</td>
</tr>
<tr>
<td>3.3.2 First-Order logic</td>
</tr>
<tr>
<td>3.4 Ontology</td>
</tr>
</tbody>
</table>

### 4 Natural Language Processing

<table>
<thead>
<tr>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Speech Segmentation</td>
</tr>
<tr>
<td>4.2 Part-of-speech Tagging</td>
</tr>
<tr>
<td>4.3 Word sense disambiguation</td>
</tr>
<tr>
<td>4.4 Parsing</td>
</tr>
<tr>
<td>4.4.1 Probabilistic Parser</td>
</tr>
<tr>
<td>4.4.2 Logic Based parsing</td>
</tr>
</tbody>
</table>

### 5 Open Source Tools

<table>
<thead>
<tr>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Stanford parser</td>
</tr>
<tr>
<td>5.2 Wordnet lexicon</td>
</tr>
<tr>
<td>5.3 Swi-Prolog</td>
</tr>
</tbody>
</table>
6 System Design

6.1 Architecture Components

6.1.1 Crawler

6.1.2 Document Repository

6.1.3 Probabilistic Parser

6.1.4 Document cache

6.1.5 Apache Lucene

6.1.6 Statistical Parser

6.1.7 Ontology Creation and Knowledge Extraction

6.1.8 Ranker

6.1.9 User query handler

6.1.10 Annotation module

6.2 Expected Work flow of the System

6.2.1 Semantic Analysis

6.2.2 Lexical Analysis

6.2.3 Another approach for semantic parsing

6.3 Query handling

6.4 Annotations

6.5 Ranking

7 Implementation

7.1 Implementation progress

8 Conclusion and Future Work

Bibliography
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Architecture of a basic search engine</td>
<td>9</td>
</tr>
<tr>
<td>3.1</td>
<td>Propositional Inference rules</td>
<td>19</td>
</tr>
<tr>
<td>3.2</td>
<td>Simple ontology of university</td>
<td>21</td>
</tr>
<tr>
<td>4.1</td>
<td>Ambiguities in natural language</td>
<td>25</td>
</tr>
<tr>
<td>4.2</td>
<td>WolframAlpha computation engine with natural language querying</td>
<td>26</td>
</tr>
<tr>
<td>4.3</td>
<td>Speech segmentation example - 1</td>
<td>27</td>
</tr>
<tr>
<td>4.4</td>
<td>Speech segmentation example - 2</td>
<td>27</td>
</tr>
<tr>
<td>4.5</td>
<td>Speech segmentation example - 3</td>
<td>28</td>
</tr>
<tr>
<td>5.1</td>
<td>Output from Stanford parser</td>
<td>34</td>
</tr>
<tr>
<td>5.2</td>
<td>Wordnet online access</td>
<td>35</td>
</tr>
<tr>
<td>6.1</td>
<td>Architecture of the proposed system</td>
<td>40</td>
</tr>
<tr>
<td>6.2</td>
<td>POS and grammatical relationship from parser</td>
<td>45</td>
</tr>
<tr>
<td>6.3</td>
<td>Graphical representation of relationship</td>
<td>47</td>
</tr>
<tr>
<td>6.4</td>
<td>XML based representation of extracted knowledge</td>
<td>47</td>
</tr>
<tr>
<td>6.5</td>
<td>Query Processing</td>
<td>51</td>
</tr>
<tr>
<td>6.6</td>
<td>Combining knowledge from different documents</td>
<td>53</td>
</tr>
</tbody>
</table>
I take this opportunity to thank everyone who helped and guided me throughout the span of the entire project and gave me plenty of opportunity to learn and work.

First of all, I express my sincere thanks to Prof Sasikumar for selecting me to be a part of the EURECA (European Research and Educational Collaboration with Asia) program funded by the Erasmus Mundus External Cooperation Window (EMECW) of the European Commission, to do my project at Mlardalen University, Sweden.

I would like to thank my Supervisor Prof. Sasikumar Punnekkat for all his help and guidance for the thesis work. His continuous support and feedback helped me a lot to improve my understanding of the subject while developing myself professionally and personally. Special thanks and sincere regards to my Examiner Radu Dobrin for his valuable assistance and help during my stay at MDH University, Vasteras.

I also express my gratitude to Prof. Stefania Costantini and Alessio Paolucci, PHD student at University of L’Aquila, Italy, for their enormous support, guidance and immense assistance in completing the project and giving me an opportunity to work and learn interesting concepts from AI and NLP.

I would also like to express my sincere gratitude to Dr. Elizabeth Sherly, Director, IIITM-K for her support, motivation and guidance throughout the entire period of my Masters course.
A large amount of digital information is spread across the internet and intranets, making it very complex to be searched and managed. This domain has been driving active research and innovation since a long time, whether it is in the field of software centric research or in terms of specialized storage and processing hardware, with a primary goal of making the information available to human beings with the minimum redundancy and highest level of precision. This goal has been further refined by the active research challenge of using the gathered information to learn new things from the existing one.

Modern search engines are highly optimized and tuned to provide fine grained information to the user. The usefulness of any search engine is measured by the relevance of the output it produces for a given query. Continuously this field has been explored since the evolution of World Wide Web to come up with a better approach to:

- Locate the most relevant information intelligently and transparently.
- Extract, process, and integrate relevant information efficiently.
- Interpret the processed information intelligently.
- Introduce Natural language processing and understanding.
- Semantic search.
- Generate new knowledge from the existing one.
Many commercial search engines have evolved to search through web documents and give the most appropriate results with minimum redundancy, becoming a pervasive part of knowledge source. In spite of being one of the major areas of research among the community of researchers and individual organizations, the current Information Retrieval (IR) systems are unable to give the expected results with complete elimination of redundancy. One solution of this problem is to be the advent of Semantic Web [1].

The major focus of research has moved in the mean time from lexical processing of web data to semantic level processing. The vision of semantic web looks forward at web pages as being consisting of textual data as annotated by semantic tags, depicting the concepts from web accessible reference ontologies. In this vision the users query will be formulated within the concepts from ontologies, instead of mere keywords, with the possibility for search engines to match the query at the semantic level and achieve higher precision.

The advent of semantic web will of course solve this problem to a large extent, but it won’t happen automatically or immediately, as the number of textual unstructured data available currently on the web can not be handled manually and at one stretch. Thus it is very important to solve this puzzle piece by piece. One puzzle to be solved is to automatically extract the knowledge from the existing web documents and further use that knowledge to create a new one along with the creation of domain specific ontologies.

Various complex approaches had been developed and many more are under research, numerous other domains had been explored to find the possible solution for scaling up the relevancy and precision level, but mostly for the internet environment only. This of course is a key success but is not available to intranet based IR systems.

Most of the IR systems are still purely database centric within the context of intranets. User’s queries are evaluated by the inbuilt search engine on the database consisting of all the information and merely presented in an ad-hoc manner without concerning about relevance.
As the Intranets are one of the major source of information among numerous enterprises and organizations, and as they are isolated from Internet, much more control can be obtained over the quality of data, relevancy thus leading towards better search results. This can not be achieved in Internet as precisely, as everyone can be a contributor in Internet and there is no central governing body to judge or control the kind of information being published. Although being open is one of the best quality of Internet in every regard, it makes it hard to control and manage. A key problem is of course the size of Internet which has led to a scenario where even simple statistics like the size of it, or number of pages in certain language, is hard to determine.

Natural language processing and Artificial Intelligence has been explored to a large extent, to solve the problem of content quality and content redundancy, but none of the works are so far near to perfection. Some of the commercial search engines have succeeded in achieving certain level of success in this respect. But in terms of Intranet based IR systems no significant achievement had been observed. More over, to take advantage of these sophisticated techniques, one has to disclose his intranet contents to these commercial search engines, which practically contradicts with the main objective of Intranet.

Natural Language processing and Artificial Intelligence had been continuously evolving since the time before Internet age. Inference logic, unification and other approaches can be utilized to build a knowledge base and to answer any given query much more precisely along with many open source knowledge contents like freebase and wordnet. These concepts can be further employed to internally represent extracted knowledge, and do computation and inference on it.

By the time semantic web vision comes to existence there is lot to do to improve effectiveness of current IR systems and the methodology of handling data. In this work we propose an IR system with the lexical processing at the core and more advanced and semantic level processing as the top layer for Intranet based environment. Further we extend the basic idea of IR systems with the help of active user interaction and feedback, in Natural language. In this work we also investigate the most accurate and effective use of linguistic wordnet lexicon ontology to process user annotations, un-
structured data and effective query rewriting [3].

1.1 Goal of the work

The goals of this thesis work can be broadly classified as:

- Come up with an indepth design of a search engine which equally weight the importance of lexical and semantic processing.
- The design should be efficient then the modern semantic search engines which use to search for query terms through “string matching” in the large set of data to get the sub-set of relevant documents, and are thus slow in response.
- Intuitive and effective way of knowledge extraction from the unstructured data.
- Effective query re-writing mechanism should be their to expand the “Noun” query words.
- System design should also consider users involvement by the annotations.
- Users annotations should be a criteria in the cumulative ranking of documents.
- Ranking of the documents should not be based on lexical approaches only, it should also consider the semantic effectiveness as well.
- System should be capable of handling existing knowledge along with the capability to infer it and learn from new knowledge.

1.2 Outline of the thesis

This thesis report is divided into several chapters for convenient reading purpose. Chapter 2 provides preliminary information on the basics of information retrieval system along with the ongoing research. Chapter 3 provides brief overview of artificial intelligence (AI) concepts, considered
within the scope of this work, while chapter 4 tries to give relevant background of Natural language processing (NLP). Chapter 5 gives an overview of the open source tools used within the context of this thesis work. Chapter 6 describes the proposed architecture and expected work flow of the system in detail. Chapter 7 puts light on the implementation issues and finally chapter 8 ends with conclusion and future work.
Search Engines

Information Retrieval is a concept which had been in existence even before the advent of Internet and World Wide Web, but the only thing which has changed, is the way of getting information. Since the last decade the Information retrieval systems has been one of the key source of knowledge and pervasive part of life. The field evolved with the need for managing and searching scientific publications and library data, changed its major use with the widespread availability of Internet among general public, further creating new challenges and opportunities in this field. Information retrieval systems became necessary part of the day to day life for many millions in couple of years satisfying their basic information needs, from general to specific, within unstructured data mainly.

World Wide Web (WWW) was the key driver in the field of IR systems. With the advent of WWW the amount of content published across the Internet grew rapidly and so does the need to manage, index and extract information to satisfy information needs with the least effort.

Widespread commercial benefit of the IR systems was another key driver of this domain, availability of better and cheaper computer hardware pushed ahead the development and research far beyond ever expected. Modern search engines are capable of searching through billions of web pages in fraction of seconds, giving the most relevant results.

Now if we get back in the past to track the history of web search engines, we can easily track down that earlier the size of internet was so small that complete list of web-servers were maintained manually, but as the number
of web-servers increase, so does the complexity to maintain the list. Then came the first recorded search engine named ARCHIE [2] with a need to search for the files available across various web-servers. This was developed by the McGill University Student Alan Emtage in 1990 at the period when there was no existence of WWW and the primary means of accessing and retrieving files was only File Transfer Protocol (FTP).

The very first crawler based search engine having capability of full-text search was WebCrawler, which allowed users to search for particular keywords in a page. This lately became an industry wide standard. With the passing of time and growing business opportunities in this domain many more search engines evolved like Inktomi, Excite, Altavista and of course yahoo with its human maintained indices.

By the end of year 1998 the commercial search engine market saw another powerful newcomer, Google[4], with its innovative idea of link analysis based PageRank[5] algorithm along with other factors for computing composite score for a web page on any given query. After the arrival of Google in the market, the commercial search engine market saw significant changes, from acquisition of smaller search engines by the bigger ones, till the extensive use of technology and knowledge to improve the end users search experience to grab the maximum market share, with aggressive use of marketing and technology.

Since last decade the research focus in this domain has moved from text-based lexical processing, to semantics of the text processing, to retrieve/produce highly relevant results. Semantics is the science of meaning, with the goal to deliver the information queried by a user rather than have a user sort through a list of loosely related keywords. Semantic search [6] is also regarded as a set of techniques for retrieving knowledge from structured data like ontologies. Such technology enables the user to specify his requirement in more detail at the query time, especially for research queries [3].

The semantic search engines are promising contribution to the existing IR systems, but it had been felt widely that the Semantic search has the power to enhance traditional web search engines, but not to replace it. A large portion of queries are navigational [3] in which user is looking for a particular website or web-page (e.g. swiss home page, youtube), and semantic
search is not a replacement for these. More over the existing semantic search engines are no where near perfection, as there are still numerous issues to be deal with. One major problem is extraction of knowledge from unstructured data. It is easy to estimate based on any available statistics that the web contains more than 80 percent of unstructured data, which is growing in parallel with the growth of WWW, with the continuous addition of unstructured data from various sources like blogs, forums and even websites. This in turn is increasing the complexity and challenges, to create ontologies from them.

2.1 Architecture and working principle of basic Information Retrieval(IR) system

All search engines have mostly the same components, but they differ in their implementation methods, search logic and tuning methods. Search engines or IR systems usually consist of several components glued together to work efficiently. Performance and relevance are of major concern. A brief idea of how the various pieces fit together is given in this section with a basic architectural diagram (see figure 2.1).

A search engines create its listing automatically based on implementation logic or with little (or no intervention) manual intervention. They crawl the web to create a directory of information. People can search through the directory created in the above process, when changes are made these engines detect and update the index based on some specific criteria to keep themself updated.

The process of Information retrieval begins with the crawling of web documents. The web documents are crawled and stored in a local repository for further processing. Crawler is a program which once given a seed URL [3], follows that Unifrom Resource Locator (URL) and retrieves the page. This page is parsed in order to extract other URLs by URL Normalization [3], these URLs are stored in the database for crawling. Any sophisticated web crawler follows a protocol called Robots Exclusion Protocol [7], to retrieve the publicly accessible web pages from any given web server. A crawler can visit the web pages regularly or after any specified time interval, this
determines how frequently information is indexed and incorporated as a search term within a search engine. The process of crawling can include distributed crawling, DNS caching and other sophisticated techniques.

In the above figure (see figure 2.1), the locally stored documents stream in for parsing and linguistic processing. The process of parsing involves parsing the documents to extract the meaningful data and remove the unwanted HTML, JavaScript and CSS tags. Linguistic process involves stemming, tokenization and other co-related activities like language detection and so on.

Stemming [3] is a process of reducing the words to their root form. Although using most of the algorithmic approaches it has been found that it is a very crude way of identifying the tokens for creating index. This approach doesn’t consider the context of the words and thus cannot discriminate between words with the same meaning. There are many efficient stemming algorithms. A very common and effective stemming algorithm for general English is Porter’s algorithm [8].
Another more sophisticated method of extracting keywords to create posting list is lemmatization [3], which rather tries to group the word with the different inflected form as a single word, giving more scope to semantics of the word. The goal of stemming and lemmatization is to reduce inflectional forms of a word to a common base form.

For instance **lemmatization** gives:

* am, are, is --> be
* play, plays, play’s, plays --> play

whereas **stemming** gives:

* cats --> cat
* blocked --> block
* making --> mak

Lemmatisation offers a better approach as a Natural language processing tool but the problem in this approach is possibly due to the limitation of currently available techniques. Stemming increases the recall hurting the precision to a large extent, whereas lemmatisation can increase precision in certain cases.

The resulting stream of parsing phase is fed into two modules. One copy is stored in the cache to help in generating result snippets (snippets of text in the query results list along with the URL to resource) and other processing (Semantic). The result snippet is used to give an idea to user that why the resultant document was given in response to the query.

Second copy of the resulting stream of parsing phase is given to indexers. These indexers create various kinds of index out of the input document, including zone and field indexes which store the metadata for each document, inverted positional indexes [3], indexes for spelling correction and other useful information. Indexes are updated when a page is modified or added. Until a page is indexed it is not available for search. Indexing may incorporate most sophisticated logic to get the most important information from any particular web page. Indexing may even incorporate semantic processing of the documents, which helps in understanding the semantic
of the words.

Once a document is indexed it is ready to be searched for any given query. User query module takes the input from the user and from the millions of entries of indexed data, finds the best result for any given query. The user query can also be given to other optional module like spell correction. Various standard approaches are followed for this very purpose. These approaches are divided in two broad categories based on isolated query terms and context sensitive query terms. With respect to isolated query correction possible approaches are:

- Edit distance calculation [3].
- K-gram [3].

Either of the two or both of them can be employed to get the result with the highest level of accuracy. In terms of context sensitive approach, where the query term is more than a single word, every word can be looked for correction by the isolated word correction approach and then by further substituting all the available possibilities among the words.

Example:

- If the submitted query is "flwe form Rome".
- And the expected query is "flew from Rome".

In a case like above every possible correction can be found for every single word in the composite query and then tried with different combination. Possible correction for:

* flwe --> flew, flea, flue..
* form --> from, fore..

Then the possible combination using above corrected words can be:
The result retrieved for each of the new composite query is then evaluated to find the higher precision on the recall set. Trying a query with different combination also results in increased processing time for a given query with increased resource utilization. Apart from these approaches another possible approach for spelling correction is using phonetic correction techniques.

Once the query is applied on indexes, the retrieved results are ranked based on the relevancy. The relevancy of the search and ranking can be measured and assigned to the result set, based on many typical logics. A common measure is the search term frequency. Some search engines consider the search term frequency along with the positional information (words appearing at the beginning in the document is given more importance).

Ranking is very important for the users, as it is extremely hard to walk through all the results to get the desired information. These ranking techniques tries to filter out the most accurate and important documents, to be given as the first set of result. Other approaches for refining the results are:

- **PageRank**: The iterative link analysis based PageRank [5] algorithm ranks any particular page based on the number and PageRank of other websites which links to the page. It was found in several studies that Link Analysis algorithms were having high precision in case of queries looking for a particular page e.g. ”Home-page of Swiss Airlines”.

- **BrowseRank**: BrowseRank [9] approach computes the importance of any given page by using a ’user browsing graph’. It considers that a user visits frequently, and spends more time on an important page. This approach is although contradictory with the fact that a large number of users spend most of their time on forums and social networking sites, causing certain limitation.
• **Trust rank:** TrustRank [10] is a link analysis technique widely used for ranking for semi-automatically controlling spams from useful webpages. Numerous websites create spam pages to achieve higher page rank within search engine results, mainly for commercial reasons. In this approach of ranking pages a small set of pages is verified manually as being of useful and trustworthy, once a set of trustworthy pages are identified the URL extending outwards from these pages are crawled considering them to be trustworthy as well.

Apart from these widely acclaimed ranking algorithms there are numerous other factors which are considered in a sophisticated search engine employed for commercial purpose. Another interesting approach for ranking pages is based on clustering. Similar websites are grouped together in the clusters forming cliques [12]. These cliques represent alliance between closely related content creators like business partners, educational institutions etc. These cliques can be further analyzed and structured together to create knowledge base for other sophisticated analysis and complex processing.

### 2.2 Challenges

Apart from numerous challenges in extracting meaningful information from the documents, IR systems face several other challenges as well (as discussed at [3]). Some of them can be listed as:

- **Problem of size:** The web-world has grown into enormous size. Hence creating the index of the whole pages in the web is a very big problem. Present day search engine can possibly index only a small subset of all the web pages.

- **Problem of consistency:** Many web-pages updates frequently, forcing search engines to visit these pages periodically and frequently, thus adding extra work.

- **Storage cost/time taken:** If we have 10 billion pages of size 10kb each then it requires 100TB of storage for storing them in local repository after crawling. A commercial search engine requires much more resources, to provide high availability, create and maintain index and to calculate query results.
• Other problems:

• The allowed queries are typically limited to search for keywords, generating many suspicious results. Better results may be obtained by limiting matches within a paragraph or phrase, rather than matching random words distributed across the entire page.

• Dynamically generated sites may be slow to index.

• Spamming and numerous other approaches to alter the search results.

2.3 Ongoing Research and future

Search engine development is progressing towards more complex approaches with the prime orientation towards knowledge extraction and computation, although lexical approach at the base. Focus of research in this domain is currently on three aspects:

• Natural Language Processing.

• Structured data (Representation and Extraction).

• Representation of extracted knowledge and computation on it.

Wolfram Alpha [11] is an interesting project with the capability of being a high performance knowledge computation engine. This project is developed by Wolfram Research and is capable of doing complex computation on structured data, rather than giving merely keyword based result. Though it is neither a lexical search engine nor a semantic one, it is a computation engine, which generates the knowledge based on inbuilt complex logic. Thus practically it can answer the questions which are not available as specific terms and phrases on the web.

Another interesting project within this context is PowerSet [13]. It is basically a semantic search engine with the capability to extract the knowledge from partially structured text of wikipeida and satisfy the query after computation on extracted knowledge. It focuses on ontological data extraction
and the capability to answer the interrogative queries starting with wh (wh-word). It uses linguistic processing to understand the query.

Other sources of structured data and knowledge include:

* DBpedia.
* Freebase.

These sources provide application programming interfaces (API) for integration with any IR system. A lot of research has to be done in the context of knowledge extraction, creation and processing, as none of the above mentioned projects are near to perfection yet. Moreover, none of them can be considered as a full fledged IR system with the capability to replace modern world search engines. They lack in several facets and thus can be considered as a supporting tool to provide more sophisticated results, working in parallel with the lexical search engines.
Chapter 3

Artificial Intelligence (AI)

Artificial Intelligence is a branch of computer science dealing with intelligent behavior as observed in human beings, and to imitate the same with the help of intelligent entities. Intelligence has been a very old discipline, but mostly among philosophers, whereas artificial intelligence (AI) is a term coined recently only and has been a very active field of study since then. Although the research in this field has been going on now for quite some time, but still there is a vast field to be explored with a challenge to understand and imitate the biological brain, the way it used to reason, infer the knowledge and weaves them all together to build new knowledge.

If all the definitions given for the term AI are combined together, then we can easily figure out four categories [14] and centers of research:

* System that think like humans.
* System that think rationally.
* System that acts like human.
* System that acts rationally.

AI and Natural Language Processing (NLP) had been interdisciplinary areas of research for quite some time to perceive the human language and to build the knowledge base for agent based systems, to give the output in natural language and process the language logically with the support of grammatical rules, as it is done by the biological machine (human brain).
The capability of human brain is extremely hard to achieve in an agent based system, but still the research had been continuously progressing towards the quest for success. Human brain immediately after getting into existence starts building/growing its knowledge, by associating the images for any word or thing it encounters, and continuously weaves the new knowledge around those images or base facts for everything.

This indeed is not possible in case of computer systems currently as they are dumb machines and are not capable of building or weaving their knowledge around any given object/image of their own. Thus they need the knowledge base along with the facts and rules with some logical reasoning, to understand, infer and learn new knowledge from it. One of the most interesting uses of AI in the context of computer science had been agent based system, which indeed is a quest to build systems with human brain alike reasoning capabilities. This quest had been refined in the present scenario to build software agents which can handle, manage and share the work and have a vast knowledge of a particular domain.

3.1 Software Agents

A software agent is a piece of software that acts and perceives in any environment on behalf of someone else, takes the decision and acts logically. These agents depict the property of self-adaptation and self-learning.

Software agents offer a new paradigm for large-scale distributed heterogeneous application development. These distributed and cooperating agents adapts and acts in a logical way proposing a new paradigm for knowledge based intelligent systems, communicating, learning and sharing their knowledge all together.

At the core of all agent based systems lies various logical and mathematical rules to make them capable of taking decision, to establish a goal and achieve it. A knowledge-base is of course at the core of any such kind of system consisting background information.
3.2 Logical Inference

Logical Inference is a process to derive any conclusion from the existing facts. It helps in deducing the relationships between sentences [14]. A sentence is considered satisfiable if there is some interpretation in some world (knowledge base in case of software agents) for which it is true, while self-contradictory sentences are considered unsatisfiable.

A logical inference system in AI is considered to be one which can deduce new facts and extends the knowledge base. This knowledge base consists of propositions holding the systems knowledge about the world.

3.3 Logic

Logic consists of a system describing the state of affairs with syntax and semantics of the language, to associate systematic relationship among sentences along with some set of rules for entailment. There are several logics studied within this context, but our focus will be on propositional and first-order logic only within the context of this thesis.

3.3.1 Propositional logic

In propositional logic [14] we depict the facts from the world in terms of symbols. The propositional logic is built of constants like True and False, proposition symbols and logical connectives like 'AND', 'OR', 'IMPLICATION', 'BICONDITIONAL', 'NEGATION' and 'PARANTHESES'. Complex sentences can be built around with the help of atomic sentences and logical connectives. Truth table can be used to define connectives and validating the sentences.

Truth tables are good for validating the propositions, but they can’t be used to reason about things the way human beings do. These patterns of reasoning are captured within inference rules for propositional logic. Given is the list of few commonly used inference rules:
These inference rules along with factual knowledge can be used to deduce the facts and capture the constraints from the real world, as propositional logic are good with the constant terms like 'True/False' rather then relationships and properties involved among real world entities, which can be better captured with the help of first-order logic (see next section):

Example:

If we have in our knowledge base a fact telling that "You can cast a vote only if you are above 20 years" in propositional logic form

* P = Cast a vote.
* Q = Over 20 years.
* P --> Q (Propositional fact)

Using this fact we can deduce with the help of inference rules whether a person is eligible to cast the vote or not(if we have the age information) as the rule above states that the necessary condition for 'P' is 'Q'.

To do systematic deduction once we have knowledge base and a goal say 'G' and asked to check whether the goal is implied from the knowledge base or not, we can push the negation of 'G' in the knowledge base and check the system if it becomes unsatisfiable.
3.3.2 First-Order logic

First order logics [14] are capable of capturing broader scope of information from the real world as opposite to propositional logic which captures small details of the world with the simple declarative sentences. It considers world to be a collection of objects with different attributes of their own along with the functions on those attributes and relationship between objects. It also includes a world of discourse over which the quantifiers (Universal and Existential) value varies.

The syntax of first-order logic consists of various elements like:

- **Constant symbols**: Represents an object in the world e.g. P, R, Tom etc.
- **Predicate**: Predicates can have one or more arguments and can be used to denote property or relation between objects e.g. Father(x, y).
- **Function**: Functions are capable of returning some value e.g. Cosine.
- **Term**: Terms refers to objects thus constants are also terms.
- **Atomic Sentence**: It is formed with the predicate symbol and terms as argument for it e.g. Friends (Tom, John).
- **Complex Sentences**: Complex sentences can be formed with the help of logical connectives e.g. Older(Tom, 20) :- ¬ Younger(Tom, 20), states that if Tom is older than 20, then he is not younger than 20.
- **Quantifiers**: Quantifiers denotes universe of discourse. They are of two types:
  
  * Universal Quantifier.
  * Existential Quantifier.

There are several methods of doing inference on a knowledge base (KB) consisting of predicates and functions derived from first order logic. For example just with the substitution of ground term ’Tom’ with ’x’ in the predicate:
* ∀x \(\text{likes}(x, \text{ice-cream})\)

We can deduce that \(\text{likes(Tom, ice-cream)}\) as 'Tom' is also there within all people denoted by \(\forall x\).

First-order logic provides a sound basic for knowledge engineering once the domain specific concepts are converted into ontologies (see next section) and predicates, and when functions and constants are identified. This indeed helps in creation of new knowledge which can be added to the existing one for continuous acquisition.

3.4 Ontology

Ontology is a structured knowledge of the world or any particular domain described in the terms which denotes contents from the world. Ontologies are derived from the facts and cover the details of various aspects and information of the objects and relationship among them for sophisticated knowledge engineering quest. It provides a standard approach for knowledge sharing and makes the people and computer agents with unlike qualities, communicate together.

![Figure 3.2: Simple ontology of university](image)

Ontologies had been researched and used extensively for enhancing the IR
systems, so that the knowledge acquired can be used to satisfy user query more effectively, rather than user collecting every piece of information on a particular topic from hundreds of web-pages [15] [16]. The pursuit of acquiring useful knowledge from un-structured data available over web had led to various open source general purpose domain specific ontologies development.

Ontology can consist of objects, attributes, concepts, relations and function from the real world with vocabulary to denote those attributes, object and relations. It may also include constraints to captures significant aspects of domain and the world. A simple ontology is depicted in the figure 3.2. This ontology shows the various entities in a university, involved in certain relationship with each other. It is also possible to have different ontology for the same domain with different vocabulary and covering different level of abstraction for the given domain.

Ontology can be either of the following type:

- **General Purpose**: As the name states this is a general purpose ontology and thus its loosely coupled and covers a broader view of the world. It often considers the terms found in several domain specific ontologies.

- **Domain Specific**: This ontology is typically focused on one domain and thus considers the objects and other artifacts as per the knowledge of that domain. For example term mobile can have several meanings in various domains, but if its considered in terms of ontology built for cellular companies, it will have a specific meaning like mobile phones or something within the same context.

- **Task Specific**: This kind of ontology usually focuses on the task at hand and thus has limited knowledge of the domain or world. These ontologies focus on application specific requirement and are local within those applications, having limited use.

Automatic Extraction of Ontology is although a challenging task with a promising outcome to generate new knowledge from them, with help of logical inference and deduction. Another problem in developing ontologies is the changing nature of information. Thus most of the things in ontology need to be updated frequently adding extra resources for the very same
purpose. It also needs timely human intervention as in case of automatic ontology extraction. Currently available technologies are not so very advanced to differentiate between two similar terms with different properties of their own.
Natural Language Processing

Natural Language Processing (NLP) had been an interdisciplinary area of research and study spanning from Artificial Intelligence (AI) to linguistic with the aim of developing formal methods and representation of natural language to permit automatic processing and communication. Its considered as a sub-domain of AI as the study and understanding of natural language requires the ability to understand the grammar of languages and do computation or manipulation over it.

The study of NLP is formally as old as the vision of robotics and intelligent agents, to communicate with them in human language. This indeed requires the computers to be able to understand and reason human language just like human beings, and be as good as they are with the computer languages. They can be much more useful if they can take the input from human beings in natural language, process it and give the output in human language or else they can automatically translate a regional language into another, process and give the result.

Natural language processing is not as easy as it seems, it’s very difficult to be parsed correctly and contains lots of ambiguities. Consider a classical sentence which is most often discussed within any study of NLP, "Fed raises interest rate half a percent to control inflation". This is a very general example of an ambiguous sentence. When we look at the sentence more closely we can figure out that the word ‘rates’ can be either a noun or a verb (see figure 4.1) so it can be "He rates high" or "Your water rates are high". Same goes with the word 'interest' and 'raises'.

24
Thus fundamentally in a language like English, with precise grammar there could be so much of ambiguity, where so many words can be either a noun or a verb and in a general case any noun can be used to form a sentence with noun as verb, its extremely hard to structure even a small set of sentences.

The ambiguities are not only syntactical there are numerous ambiguities in the semantics as well. Consider an example sentence “Can you put that can?” this kind of sentence can drive crazy even human beings, so what about computer systems, who don’t have any idea which ‘can’ is what? This is just a very small example of complexities involved in dealing with natural language, there are many more as well.

NLP gives a broad scope of research within itself; few of them can be listed as:

- Knowledge representation
- Reasoning
- Computational Linguistics
- Machine Learning
- Statistics

One classical problem within the context of Information retrieval is named entity recognition and extraction from the unstructured textual data available in sufficiency over web. This indeed is a challenging task to accom-
plish, from identifying the syntax of the language to understand it semantically and do computation on it. Consider a scenario where computers can identify the entities and their attributes and relationship they are involved in from millions of web-pages and create a knowledge base containing all the information. This can be a major breakthrough in the era of knowledge engineering and the way information is retrieved and presented.

Some of the information retrieval systems have succeeded in accomplishing certain level of success within this context, they can identify and understand the queries entered in natural language and process them as in the manner needed. One such interesting example is "WolframAlpha"[11], which is a computational knowledge engine and offers a more advanced way of computing the queries with the existing knowledge. You can pose a query like 'What is the date today?' or may be "what is the temperature of Vasteras?", and expect a logical answer (see figure 4.2) rather than couple of links pointing to various pages with the term 'Vasteras' or 'Temperature', which is common with most of the IR systems.

![WolframAlpha](image)

Figure 4.2: WolframAlpha computation engine with natural language querying

There are a lot of subtasks within the context of linguistic processing, some of them can be listed as:
• Speech Segmentation
• Part-of-speech tagging
• Word sense disambiguation

4.1 Speech Segmentation

Speech Segmentation is the process of identifying and dividing a given text in tokens according to a given grammar, it’s the process of identifying the boundaries of the words, syllables and phonemes in natural language (see figure 4.3). In any natural language, the meaning of any given complex sentence can only be determined once the distinct words or tokens are identified. Once all the tokens (lexical) are identified then only all of them can be combined to understand and derive semantics of the sentence.

Figure 4.3: Speech segmentation example - 1

Figure 4.3 can be considered as the easiest problem for speech segmentation, the complexity can increase unexpectedly in certain cases (see figure 4.4 and 4.5).

Figure 4.4: Speech segmentation example - 2

Thus, accurate lexical segmentation depends on the context and semantics which in return requires advanced pattern recognition and artificial intelligence techniques to compensate human knowledge and experience of natural language.
4.2 Part-of-speech Tagging

Part-of-speech Tagging is the process of marking each token identified by speech segmentation or text segmentation process, with a tag representing word category like noun, verb, preposition, adjectives, adverbs etc. This process is very similar with the process carried out at elementary school in English grammar class. This may be an easy task for someone who is expert with grammatical rules, but not an easy job for computer systems. Dealing with such task requires the knowledge of semantics and context of the language along with the tools to deal with the ambiguities.

For Example: "The man shoots a bird"

Sentence (The man shoots a bird)
* Noun phrase (The man).
Determinant (The)
Noun (man)
* Verb phrase (shoots a bird)
Verb (shoots)
Noun Phrase (a bird)
Determinant (a)
Noun (bird)

This in fact is represented with the help of parse tree which represent the syntactic structure of the sentences based on the grammar of the language (see figure 4.1). This tree represent sentence as the root node, various part of speech as branch node and leaf consisting of various terms from the given sentence.
Difficulties arise when the words are encountered, which can be denoted by more than one part of speech. Another important aspect is that if at elementary school students distinguish among 8 frequently used parts of speech (noun, verb, adjective, preposition, pronoun, adverb, conjunction, and interjection) in PoS tagging this number goes from 50 to 150 for English language only.

A general method of dealing with this process has been self-learning POS-Taggers. These taggers are provided with manually tagged repository to gain knowledge of the most frequently encountered sentences and their tagged output. The first annotated corpus was the ”Brown Corpus”. Each word was manually annotated over many years and contains a collection more than 1,000,000 frequently encountered English words.

There are several approaches to PoS tagging, some of the frequently encountered ones are:

- Dynamic Programming.
- Rule-based.
- Neural-based. Artificial Neural Networks (ANN).
- Logic based. Definite clause grammar (DCG) and variants.

### 4.3 Word sense disambiguation

This is a process in computational linguistic to determine the meaning of the word in a given context. Once a word is given and its possible meaning for different context from a lexical database like WordNet [17], the task is to identify which meaning fits with which given sentence. For example for the word 'POP' wordnet lexicon gives following meaning which can fit in different context of the sentences:

- Pop - an informal term for a father.
• Pop - sweet drink containing carbonated water and flavoring.
• Pop, popping - a sharp explosive sound as from a gunshot or drawing a cork.
• Pop music - music of general appeal to teenagers; a bland watered-down version of rock’n’roll with more rhythm and harmony and an emphasis on romantic love.

A lot of research had been carried out in the domain of linguistic to overcome this problem, and its claimed by many of the approaches to claim above 90 percent accuracy, ranging from dictionary based methods to supervised machine learning approach based on corpus of hand-annotated examples.

4.4 Parsing

In natural language processing, parsing is the process of construction of a syntax tree from a stream of tokens generated from lexical analyzing (tokenizer) or speech segmentation. With respect to linguistic processing a parser is the tool which does the parsing of input sentence by validating the input for correct syntactic structure and building a data structure, usually a parse tree. Input to any given parser is the sentence as tokenized stream along with the grammar of the language, while output can be a parsed tree along with the semantic relatedness of the tokens in the context of the sentence.

4.4.1 Probabilistic Parser

Probabilistic parsers [18] [19] uses knowledge of human language gained from manually-parsed sentences and rules with a probability associated with them, to determine the possible parses of any given sentence, locating the most closely possible parse to produce the most likely analysis of the new sentence.

Thus any probabilistic/statistical parser focuses on three key steps:

• Finding possible parse.
• Assigning probabilities to possibilities.
• Select the best among them.

The simplest mechanism to achieve this is based upon ”probabilistic context free grammar” or PCFGs. PCFG is a context free grammar in which each rule is assigned with a probability based on the frequency it appears.

Example:

* S - NP VP (1.0)
* VP - Verb NP (0.8)
* VP - Verb NP NP (0.2)

Once the probability for every given rule is assigned, the probability of a parse can be found by multiplying the probabilities for each of the rule. For example with the above rules and probabilities assigned to them we can determine the number of NPs expected while deriving VPs is 0.8 x 1 + 0.2 x 2 = 1.3.

### 4.4.2 Logic Based parsing

Logic based parsing methods offers simple and effective approach to parse a wide range of natural language. The simplest approach is with the help of definite clause grammar (DCG). DCGs are more closely related with AI and Prolog, a logic or declarative programming language. The DCG provides a way of parsing natural language in the form of theorem proving rather than other more complex approaches [20]. Thus it provides an intuitive way of extending and defining new theorems (grammar) for a lot other natural languages other than English.

Sample Grammar using DCG

```
s --> np, vp.
np --> det, n.
vp --> v, np.
vp --> v.
det --> [the].
```
This approach provides techniques for computing semantic representations of natural language along with the capability to do inference on it. Vocabularies along with the facts and rules are given to deductive first-order logic systems [citation needed representation and inference for natural language vol.1], which are sound and complete, to get the desired output. The DCG can be extended into Augmented DCGs or semantically augmented DCGs (SEDCG) [21] to capture more accurate information like gender, singular and plural words.
Open Source Tools

We had been exploring several open source tools within the context of this thesis work. Though it’s very hard to describe the properties and features of all of them in their entirety, but still we try to summarize the basic idea of each of them in the subsequent sections. For further knowledge about these tools, their official documentations can be referred at any time.

5.1 Stanford parser

Stanford parser [22] is a statistical parser which uses knowledge taken from manually-parsed sentences and tries to use this knowledge for identifying the various relationships in between sentences. This parser is available for download under GNU GPL v2 license, thus can be used for research and free software projects.

This parser is capable of processing plain text (ASCII) input and generates a parse tree, a context-free phrase structure grammar and a dependency representation or grammatical representation, per sentence. By default it uses probabilistic context free grammar to parse. Currently it supports few other languages like Chinese and German apart from English and can be trained using syntactically annotated data like Penn tree bank or ICE-GB.

The parser can both be invoked from command line or through Java API, and is having proper Javadoc documentation available as part of the package. Thus it’s easy to incorporate and utilize the parsing capability of the parser. This parser consists of a tokenizer, a part-of-speech tagger and
grammatical relationship generation module. Once trained on a properly annotated data, this parser is capable of giving high level of accuracy, as natural language is highly ambiguous. Once invoked from the command line the parser gives the output for any given input which can be viewed in the figure below (see figure 5.1), same can be achieved from Java based invocation (as stated earlier).

In the figure 5.1, part-of-speech output for the given sentence is displayed in area marked within yellow color, while the one marked blue shows the grammatical dependencies within the sentence. These grammatical relationships provide adequate knowledge regarding the textual relationship and thus give an easy approach to identify the subject, object and other related information.

There are 55 such binary relationships in the current version of the parser, and can be studied in the manual provided along with the parser. Few of them can be listed as:

* Nsubj( x, y)
5.2 Wordnet lexicon

Wordnet is a lexical database of English words. In this lexical dictionary, all the concepts are defined in terms of synonym called synsets. These synsets are further organized in 'is-a' relationship with the help of hypernymy/hyponymy and semantic relationships like 'part-of/has-part' with meronymy/holonymy. Thus it is regarded as a database, which is based on the facts i.e. conceptual and lexical nature. This kind of ontology can be very helpful to a wide variety of information processing needs as it can be used effectively for filling up the gaps in structured data processing and creation of ontologies.

Wordnet lexicon ontology is freely available for download. A sample output from the web-based interface to the wordnet is displayed in the figure 5.2.

Figure 5.2: Wordnet online access

5.3 Swi-Prolog

Swi-prolog is an open source tool supporting declarative programming language Prolog [14]. Prolog stands for Programming in Logic and had been widely used among AI research inspired by logic. It has been used for logical theorem proving along with logical approach to natural language
processing with definite clause grammar. It is more closely related with semantics of logic then the syntactic, procedural or object oriented approach followed by most of the programming languages, widely used for application development. Swi-prolog also provides a rich set of features like ODBC support, and interface to python, C, Java and other programming languages. Thus it’s a convenient task to embed the logical processing capability of prolog in any general software application.

Logical programming is more about proving theorem based on the existing facts and logic. Prolog basically uses unification, a concept from AI as a built-in term-manipulation method which passes the parameters, manages data structures and returns results. Its basic control flow mechanism is backward chaining or backtracking. A prolog program consists of facts and rules same as predicates and sentences in first-order logic, thus providing a platform to deduce the complex facts using first-order logic.

Example:

Fact Duck(Donald).
Fact Male(Donald).
Rule Maleduck(x) :- Duck(x), Male(x).

A rule consist head and body, where head is satisfied when the body does. Thus in a rule like the one given above, head will be satisfied only if the given argument is a duck and is male. The output from the prolog can be achieved by posing the queries like:

?- Maleduck(Donald).
   Yes

Prolog heavily relies on the facts and rules within its knowledge base. Thus if we can extract facts from the unstructured data, we can use it to create our knowledge base and do logical inference on it.

5.4 Apache Lucene

Apache lucene [23] is an information library which had been under development and refinement for now quite some time and thus it’s a mature open
source tool for indexing and searching through the documents. Lucene provides an easy integration and implementation approach to provide basic search engine capabilities and thus had been widely used for search engine technologies and site-specific search modules.

Though it have the capability of indexing the documents and searching through it, it’s not a full-fledge search engine as it lacks crawler and many other sophisticated components needed for a search engine. It is capable of lexically processing the documents which are in plain-text format. Though there are supporting libraries to help parse other documents formats for the input to lucene.

Lucene has a fast and scalable API for full text indexing and searching capability. It can be used efficiently to add search capability to any application consisting of data which can be converted to textual format. It indexes the input text in the form of inverted positional index and stores several other information’s like, term position, frequency in a document, and so on. This is a highly optimized process resulting in fast extraction of information from the indexed text.

Lucene also provides several sophisticated and standard approach for query handling including Boolean query, Phrase query, fuzzy query etc. along with various inbuilt text analysis tools like stemmer and stop-word remover for filtering out the tokens from the given text. It also provides methods to sort and rank the result based on term frequency-inverse document frequency (tf-idf) [3] [23], which is used to measure the importance of a word for a given corpus.

As lucene is highly capable of handling the lexical processing of the documents within a corpus, it gives a broad scope to the development and research of technologies dealing with semantic processing of the same, providing a lexical base at the core of IR system.
System Design

Architecture of the proposed system comprises of several key components working all glued together. Major concern is given to extraction of knowledge, annotation based ranking and inference on the extracted knowledge to deduce new concepts from the existing one.

6.1 Architecture Components

In this section we will give a high level overview of the complete system (see Figure 6.1). The overall system design consists of 11 tightly coupled components working in parallel with each other to hold the data, parse, process and handle the annotations and user queries along with the components to extract the knowledge, indexes and rank the results. In subsequent sections will discuss the expected working of the whole system in depth with specific details on each system components.

6.1.1 Crawler

First component of the system is a web crawler which crawls through the pages within the Intranet and stores them in local repository. It crawls the entire intranet for the first time and subsequently crawls only those pages which are newly added modified in any web-server within the intranet.

The crawler maintains a list of pages available in the Intranet and keeps it updated with the help of ‘infotify’, available in linux kernels 2.6.13 and
beyond. It’s an inode based file system notification technique which generates events in case of any changes within the file system. This approach helps in re-indexing the files that are modified, deleted or recently added within the web-root directory without the need of re-indexing the whole server root again and again.

In case of dynamically generated pages a database trigger can be used for event notification to the crawler as ‘inotify’ can deal with the file-system events only.

In case of any event either from the ‘inotify’ or database triggers, crawler validates its information of URL’s and crawls new or modified pages and discards the already crawled pages from its database. Crawler maintains a database of the crawled URL and the path to the pages in the document repository.

6.1.2 Document Repository

The document repository is the second component of the system which stores the crawled documents for further processing. This repository can either be a database or a file system, it depends upon the need and number of documents been crawled as there is a restriction on number of files a file system can possess. Thus to overcome this problem either a database is used or a virtual file system like Hadoop file system (HDFS) [24], which can accommodate large number of files. The only job of this component is to hold the un-processed documents and forward it to the parser. Within the context of this thesis work we have considered to use linux file system as the number of documents is comparatively less for study purpose then the actual scenario.

6.1.3 Probabilistic Parser

Parser is the next component which automatically parses and extracts the useful data from the web-pages after removing un-necessary tags i.e. HTML, JavaScript, CSS, etc. The main purpose of the parser in to parse the web-documents and extract the actual data in plain-textual format, although it is not an easy task to develop a parser which can determine the millions
of possible errors within HTML tags and handle non-ASCII characters. Another most interesting problem for a parser is language detection. The architecture of the whole system is designed as per the idea of web-pages in Standard English language. Thus Non-English terms may get un-noticed while processing, and may even cause certain problems. The output of the parser is streamed into two components (viz. Document cache and lucene) for further processing.

6.1.4 Document cache

First component to acquire the stream from the parser is a document cache which holds the filtered data for the process of knowledge extraction. This in fact is also used for generating result snippet and result summary. The
second component to receive the stream is lucene IR library. We have considered using a relational database for this very purpose for now, but the very same purpose can also be satisfied with the plain files on file system. In our design, document cache also holds the extracted knowledge, facts and rules along with the annotation information. We considered the use of database to keep the data at one central repository for analysis and control the redundancy. Analyzing each of the file for every single web-document can be a cumbersome task.

6.1.5 Apache Lucene

Lucene indexes the parsed documents and creates an inverted index for faster search. Lucene creates an inverted positional index of the terms and in sorted manner. This index is created and maintained by the lucene for faster retrieval of documents based on particular keyword (lexical searching). This index contains the information for every word such as documents information in which a word appears, frequency of any particular word in the document and positional information for the word in every document. Due to the effective and highly optimized performance of lucene IR library for indexing and searching (lexically), we can focus on other techniques for better knowledge oriented search experience, especially semantic side.

Lucene also maintains an inverted index of the wordnet lexicon for faster retrieval of the synonyms. It uses the wordnet prolog package to build its index, which can be queried with a particular word, and returns an array of all the synonyms for that particular word. Lucene in fact also provides various sophisticated ways to handle different kinds of user queries like

- Term query
- Range query
- Boolean query
- Prefix query
- Phrase query and
- Wild Card query.
The various methods in hand allow querying the index based on particular kind of query need.

6.1.6 Statistical Parser

This is the parser for extracting semantic information from the parsed text. This component gets the data from the document cache and tries to figure out grammatical structure of the sentence, to determine the verb and its subject and object. We have considered statistical parser from Stanford University for this very purpose. This parser uses Stanford typed dependencies to provide grammatical relationships in a sentence. The use of parser helps in building a knowledge base for doing inference on it. And thus getting the part of speech tagging along with the Stanford typed dependencies of the sentences.

Prior to the parsing step, extraction of semi-structured data can be done with the help of template based extraction techniques. This approach considers that most of the web-sites (dynamically generated) follow a pattern or a base template to produce the data on web.

6.1.7 Ontology Creation and Knowledge Extraction

This component takes the output stream from the Stanford parser consisting of part-of-speech tagging and Stanford dependencies for every sentence in a page, and tries to associate relationship in between various sentences within a page to produce ontological information after analyzing the syntactically and semantically. The extracted ontological information is used to populate the knowledge base for inferring the new knowledge from the existing facts. The working of this component is described in subsequent sections in more detail.

The extracted knowledge is saved in the database (document cache) in the form of XML along with the inference rules and facts for every single page to infer.
6.1.8 Ranker

This is the component which ranks the result set generated for any particular query. The result based on lexical search from lucene IR library is processed along with the ontological information and facts to rank the pages to get the best result satisfying any given query. It basically uses two techniques for doing so.

- Annotations given by the user for any page.
- Number of Ontological relationships satisfied by that document and importance of those relationships (explained in more detail later (section 6.5)).

6.1.9 User query handler

It takes the user entered query and re-writes the query to get the most out of the documents. Within the context of this thesis we do not consider the fact that many user entered queries needs to be checked for spelling and other corrections. This module indeed takes the help of wordnet lexicon to re-write a given query term in every possible form for that query.

6.1.10 Annotation module

This module takes the annotation given by a user in a natural language form and forwards it to the language parser to extract the various relationships in between the verb and object. We assume that the subject of any annotation is that particular page only for which it was given. This information is saved in the database as meta-data for that page along with the detail of the person giving the annotation. This module permits a page to be annotated only once, by any given user.

6.2 Expected Work flow of the System

In this section we describe the overall working of the system with all the components mentioned in the previous section glued together. We do not consider the crawling of the document in this as we assume this process to
be the standard one. We even do not consider many of the performance related issues as our main focus is on the quality of the search result, and more over modern day computer systems are capable of handling vast amount of data and complex computations. But any how integration of optimized lexical processing of lucene with semantic processing ensures that the system will be faster than any existing semantic based search engine as rather than doing string matching on the whole set of documents we are taking out a subset of documents lexically and then processing them to get the information they hold.

6.2.1 Semantic Analysis

Semantic analysis of the document tries to figure out the main component of the sentence like verb, subject and object. Named-entities like place and person name also plays a major role in ontology creation. parsed document from the document cache is fed to Stanford parser to identify and extract the grammatical relationship between the sentences along with part-of-speech. The grammatical relationships who emphasize more on noun and verb are of major importance within this context. Part-of-speech (POS) tagging is also very useful in determining certain qualities.

We took random text from the Wikipedia article on Albert Einstein for the verification of the concept we are focusing mainly. The text is represented in the figure 6.2 along with part-of-speech representation and grammatical dependencies.

From the parsed text representation of the text we can easily identify that the first sentence is about ’Albert Einstein’ which is tagged as NP (proper noun) within the POS representation and then in more detail with the tag NNP (singular proper noun), at the beginning of the sentence only. Which is further represented by the grammatical relationship ’nn(Einstein-2, Albert-1)’, which specifies that there is a noun compound modifier. A noun compound modifier of NP is a noun that modifies the head noun [Stanford dependency manual]. This information along with NNP is sufficient to determine that the subject of this sentence is ’Albert Einstein’.

Further when we analyze the sentence we can see that there are two terms denote by VP (verb phrase) and further categorized as VBD (verb, past-
tense) and VBN (verb, past-participle), the same is represented in the 2nd statement of grammatical relationship as 'nsubjpass(born-4, Einstein-2)' . The relationship 'nsubjpass' represents that the second argument is the subject of the passive clause. The information from VBD can be useful in determining that the sentence is trying to denote something in past while VBN along with the 'nsubjpass' can be easily used to determine that the verb for the object 'Albert Einstein' is 'born', of course with an additional information of past tense in the form of 'was' or VBD with the supporting statement 'auxpass(born-4, was-3)'.

If we analyze rest of the sentence we can get two more information on the verb 'born' from the grammatical dependencies 'prep_in' and 'prep_on' (prepositional modifier 'in' and 'on'). Prepositional modifier modifies the meaning of the verb, noun and adjective. Now if we use the wordnet lexicon to process all the noun words, it’s easy to determine that 'Germany' is a place and 'March' is a month, which is in between February and April. More generally if stated, 'in' is used to denote something which is enclosed or inward, while 'on' is can be used to represent relationship like Located at, attached to, In contact with, on top of, On the topic of, etc.

Lastly the information about 'March' can be combined together with the help of two relationships (num) to determine that '14, 1879' is trying to modify the meaning of the NP 'March' and is thus associated with it.
Now if we look at the second sentence, we can notice that its first two terms are marked as NP and subsequently as PRP$ and NN, where PRP$ denotes possessive pronoun and NN denotes noun which can be used for singular or mass, along with the relationship between them ‘poss(father-2, his-1)’ which shows relation between the head of a NP and its possessive determiner (possession of something). Thus it can be concluded easily that the subject is not present in this sentence, so the last extracted subject is regarded as the subject for this particular sentence and the information and the NN (father) can be considered as a relationship between the subject and the object.

Looking ahead at the POS we can see that there is a NP enclosing two NNP’s (NP (NNP Hermann) (NNP Einstein)), just like the way we had in case of first sentence. The same way we can determine that this is a name/noun. The relationship between this noun and the verb, which we identified earlier (father) can be seen in the grammatical relationship ‘nsubj(Einstein-5, father-2), where ‘nsubj’ means nominal subject. Hence this is the object of the relationship father in the context of the given example.

Last two grammatical statements clearly specify the relationship between the Einstein, salesman and engineer. The same way (as for second sentence) if we analyze the third sentence we can get the relationship ‘mother’ subject ’Albert Einstein’ and object ‘Pauline Einstein’.

The extracted knowledge can be represented in a graphical or structured textual format (see figure 6.3 and 6.4). This is the process followed for every single document in the corps to identify and extract meaningful structured data. Wordnet lexicon can be used to extract the synonyms for the relationships, this approach can be effectively used to represent and minimize number of relationships denoted by the same term like: dad, daddy, father, padre, male parent. Constraints from the real world can also be identified and included to create the most accurate information base.

The structured data (figure 6.4) can be easily used to infer new facts. Some of the relationships among entities can automatically be captured within
prepositional logic and first-order rules. Wordnet and other additional ontologies can be effectively used within this context. For Example we can easily generate facts and rules from the above example like:

- maleparent(albert_einstein, herman_einstein).
- femaleparent(albert_einstein, pauline_einstein).
- parents(X,Y,Z) :- maleparent(X,Y), femaleparent(X,Z).

And pose a query like ”parents (albert_einstein, Y, Z). ”, to get the output like:
Y = herman_einstein
Z = pauline_einstein.

This deduced knowledge can be stored in knowledge base as part of continuous learning process.

6.2.2 Lexical Analysis

Lexical analysis of the data streamed from parsing and linguistic processing is handled by the Apache lucene IR library. This library analyzes and converts the input text into stream of tokens, these tokens are further passed through stemming and lemmatization process to get the base form of the word. Then lucene library makes an inverted posting list/inverted index [3] of the words along with the other information’s. Inverted index is a data structure used widely in information retrieval systems for faster retrieval and lookup for any given keyword within a set of documents. Inverted index is called so as rather than having document as the keys to create posting list, keywords are used as keys. An example of the inverted index entries, where the words like ‘cat’, ’eat’ and ’rat’ are keys is given below:

- cat: (document-1; position 4, 9; frequency 2) (document-5; position 23, 6, 8,..; frequency 8)
- eat: (document-1; position 2; frequency 1)
- mouse: (document-4; position 12; frequency 2) (document-1; position 6; frequency 1)

Though lucene manages the process of index creation in a more sophisticated way as this is one of the major function in lexical analysis of documents, for example rather than using string based representation of keywords/terms a numerical representation is used for efficiency. Lucene also manages the process of index merging or posting list merging of its own. Once the index is ready the lucene library searches through the inverted index for any given query word either by directly mapping or searching the keyword in the index or using the wordID(a logical numerical value assigned to a particular word) to get the posting list. If the query contains two or more words then the posting list for all those words are merged by simple Boolean operators.

Example:
• cat: (2, 3, 4, 7, 8)
• rat: (1, 4, 6, 9)
• chase: (2, 4)

Once we merge these three posting list we can get:

• cat AND rat AND chase: (4)
• cat OR rat OR chase: (1, 2, 3, 4, 6, 7, 8, 9)

We can easily determine the documents containing any particular keyword or all of them. Positional information and other information stored within the posting list can be helpful in solving the phrase queries [23] and in certain cases stop word removal based on the high frequency of any word in a document.

This library is highly effective in processing large sets of data for creating index and searching within them for specific term (lexically) and thus does not give any importance or concern to the context and semantics of the document. Thus it handles all the documents as mere stream of tokens glued in it.

6.2.3 Another approach for semantic parsing

Within the context of this thesis work we studied the possibility of logical approach for parsing natural language. The logical approach is based on DCG rules (see 4.4.2) which give an intuitive way of handling properly structured natural language sentences.

DCG based parsing approach was found to be effective in case of properly structured English sentences, for which DCG rules can be created easily. But as the natural language is filled with ambiguities and there are numerous ways to form sentences with the same contextual information, there is no way of capturing the semantic relatedness among all of them with DCG. Thus it works more closely based on the proper structure of the sentence and the rules to handle them.

This in turn leads to a situation where we need to write rules for every possible structure in any given language to handle most of the available
sentence, and this is a challenging task. Apart from this, it is very hard to handle a situation where a noun can be used as a verb anytime, and there is no way of handling such information until unless DCGs are also capable of dealing with the context of the words. Thus logical approach can itself not handle the complete parsing of a natural language without some way to understand, identify and extract the contextual information.

DCGs can be better used with some way of recognizing the pattern in natural language along with inbuilt grammatical rules for the language, to effectively handle natural language, as this is the very same way by which human brain use to handle natural language.

6.3 Query handling

The user query handler takes the given input from the HTTP request and forwards it to the statistical parser to extract the important information’s like verb, noun etc., though noun words are given more importance. This process also discards the frequently occurring terms from the query like ‘the’, ’a’ etc., which are of least information within the context of the query. This process can be further extended with the substitution of synonyms for the noun terms. Expanding verbs can lead to un-necessary growth of recall and can hurt the precision in some cases, returning a lot of un-necessary documents.

Once the query is processed to extract necessary keywords eliminating stop words, the query is passed to lucene index (see figure 6.5) for identifying the documents matching the keywords. Lucene does a lookup in its index to identify the documents containing the terms or phrase and returns the documents information passes it for semantic analysis.

The documents identified by lucene lexically are checked within document cache to get the ontological information which they hold. The information possessed by all the documents returned by lucene is identified along with the facts and rules they can provide. This ontological information is used to link the set of documents returned by lexical processing, based on their semantic relatedness, and group the most closely (semantically) related documents. Once all the ontological information and facts are retrieved,
they are combined together to generate a result summarization and rank the most suitable document in response to given query. Result summarization is the knowledge extracted from multiple pages for any given query term.

6.4 Annotations

Users given annotations is taken in their natural language form and processed for the semantic relationships between the words. In this processing of course the subject is always that very page for which annotation is given. The major focus is to identify the verbs and adjective phrases, which holds the actual information given by annotations. These verbs and
adjectives with their synonyms are grouped together to categorize the documents, and to identify whether a document falls in the category of good, normal or bad. This concept can further be extended to capture other qualities like whether the document is for experts, intermediate or beginners and a value can be assigned to each of the category like:

- Good - 0.57
- Normal - 0.28
- Bad - .14

This can be used for cumulative ranking of the documents. Synsets from wordnet is very useful in identifying the closely related words and grouping them in their respective categories. The category based ranking of the web-documents or query result will let the users vote for the best page. User participation is highly effective in self-learning the importance of the documents for queries.

In an environment like intranet it is also possible to modify a query based on the annotation given by any particular user. Where a user can search based on annotations on documents given by any specific person. This can be very effective in fine-tuning the search results and limit the scope of interest to a subset of originally returned documents.

### 6.5 Ranking

Ranking of the returned documents is a three step process. In the first step the documents are checked for their term frequency-inverse document frequency (tf-idf). This is done with inbuilt mechanism provided within lucene. The tf-idf scoring sorts the document based on number of times a particular query term is occurring within a document. This statistical measure increases the weight of the terms that occur rarely and diminishes the weight of very frequently occurring terms.

Once the documents are ranked based on their tf-idf score the resultant set is ranked based on the annotations on them. Annotations are considered a nominal factor and add a bit of score in the cumulative ranking, until the
query doesn’t specify clearly that it’s looking for the document annotated by the particular user. In this particular case annotations are weighted more in overall scoring.

Final scoring of the documents is based on the relationship graphs they satisfy. This ranking considers the semantic involvement of the document in the context of the queried term. It’s often seen that for a given two word query the search engines return the pages where one term is encountered in one paragraph and another one in the next or so on. Thus our approach considers the semantic relatedness of the whole document including all the paragraphs within it.
Usual query terms appear in the paragraphs which are semantically different from each other like one paragraph may consist information about someone’s life and within the same document another paragraph may be consisting some other information which is totally irrelevant with the previous part or vice versa, but still is returned for the given query by any lexical search engine.

To solve this problem semantically we use the ontological information extracted from the parsers, we combine the entire relationship tree from different sets of document with the queried entry (especially noun terms) at the center or say root of the tree. For example in the figure 6.6 ’Albert Einstein’ is the root of the derived tree and Document-1, 2 & 3 are combined together.

Now we capture the information of the direct relationships from the root node and the next level of the relationships (immediate child of the root node) and the relationships that are occurring frequently in a large number of documents. These frequently occurring relationships are used to derive result summarization and the document satisfying maximum number of the relationships from the set of relationships retrieved after combining different layers of tree, is rated high.

For the two word query or anything beyond that, we calculate the branch distance of the terms within the combined layer of semantic relationship. The document having shortest path from root node to the next term and satisfying more number of semantic relationship is considered of more importance.
Implementation

Implementation of the proposed system is a challenging task due to lots of ambiguities involved in the process of parsing web pages. Every single component including the main component like ontology and knowledge extractor is to be implemented in Java programming language, due to the various feature and flexibility provided by it like platform independence and a large number of API’s for integrating various open source tools used within the context of this work.

Crawler can be easily implemented using Java as there are enough libraries to handle and manage socket programming along with effective exception handling management. There are several open source implementations of the crawler as well, which can be used to implement as effective web crawler. One such open source crawler is httrack with lot many features for effective web crawling like crawling based on domain, URL or free crawling, with the options to update already crawled websites.

Lucene can easily be integrated using Java as there are inbuilt and easy to use API to handle everything within it. There are several ways to handle web-pages and other document format to extract the plain-text data. Else just to handle html documents a parser can be written using Java regular expressions. Lucene also gives a way to make an index of wordnet lexicon to retrieve the synonyms immediately. Prolog version of Wordnet can be downloaded for this very purpose.

Same is the case with Stanford parser which provides Java based interface for using it. There are several classes and methods within the package
”edu.stanford.nlp” to invoke the parsing from Java and get the result their only. The only thing is to add the jar package within the code and environment.

Components like ontology extractor and ranker needs to be given more attention during implementation as these are some of the main components within the context of this work. But this can also be implemented easily with a bit careful attention and planning.

7.1 Implementation progress

So far the implementation of lexical part has completed by setting up a lucene IR library. We used httrack web-crawler on domain www.iiitmk.ac.in to collect the test data, following Robots Exclusion Protocol. This data contains only web-documents with html extension on it. The data(web-documents) is parsed with the help of a custom built parser implemented in Java using Java regular expressions, to extract the useful text discarding various un-necessary tags within the web-pages. The stream of extracted data from this parser is forwarded to lucene and one copy is saved in database.

Lucene library creates index of the test data and search through it for any keyword and returns the path to documents containing the word/s. Lucenes inbuilt class ”IndexWriter” provides the necessary methods to easily create the index or add new documents to the existing index, while the ”IndexSearcher” class provides the methods to search for an entry within the index.

The parsed data saved in the cache (databases) is forwarded to statistical parser for POS and grammatical relationship extraction. The output stream is then saved in the database for every single document to extract the ontologies from them.
Conclusion and Future Work

This thesis work presents architecture for an advanced search engine with various sophisticated techniques to deal with unstructured web-documents. The approach to handle unstructured data lexically and semantically both, gives a promising approach for an advanced search engine for intranet or domain specific deployment with an opportunity to extract and learn from the existing data.

We can easily evaluate the proposed design on the criteria (goals) set for this thesis work. The proposed system considers the faster and highly optimized performance of lexical search engine to retrieve the effective documents for any give query and then does the semantic analysis on that sub-set of documents to give the most accurate information. This in turn increases the response level as compared with any other existing semantic search approach.

Combining semantic capabilities with the lexical helps in achieving higher level of accuracy and performance for research queries and navigational queries both. The system also takes into account the active user feedback in natural language form and uses it to rank the usefulness of the documents. Ranking of the documents is further fine tuned with the help of relationship based ranking approach, which considers the semantic involvement of the documents for any given entity (as keyword).

The system also gives an intuitive methodology to extract the factual knowledge from the unstructured data and thus the possibility to deduce new facts/knowledge from it. Thus making the system capable of learning con-
continuously, and grow its knowledge base with time.

There are several interesting directions for future work including complete implementation of the proposed architecture, so that the concepts discussed within this thesis work can be evaluated in terms of accuracy and effectiveness of search results. Another direction of work can be developing the algorithms to identify and extract the useful information/ontology from the statistically parsed documents, and implement it.

Last but not the least, most intuitive and interesting direction for future work is to develop methodology to combine the lexical processing of documents and semantic processing, at the index level. If we can combine these two approaches so that any highly efficient data-structure like inverted index can hold the general information of keywords along with their contextual information, we can push the research in this field to a new level by attaining higher level of performance.
Bibliography


