Smart Search Engine For Information Retrieval

Leilei Sun

A thesis in part fulfilment of the requirements of the University of Durham for the degree of Master of Science by Research

Word count: 17353
Submitted: 24 June 2009

Abstract

This project addresses the main research problem in information retrieval and semantic search. It proposes the smart search theory as new theory based on hypothesis that semantic meanings of a document can be described by a set of keywords. With two experiments designed and carried out in this project, the experiment result demonstrates positive evidence that meet the smart search theory.

In the theory proposed in this project, the smart search aims to determine a set of keywords for any web documents, by which the semantic meanings of the documents can be uniquely identified. Meanwhile, the size of the set of keywords is supposed to be small enough which can be easily managed. This is the fundamental assumption for creating the smart semantic search engine. In this project, the rationale of the assumption and the theory based on it will be discussed, as well as the processes of how the theory can be applied to the keyword allocation and the data model to be generated. Then the design of the smart search engine will be proposed, in order to create a solution to the efficiency problem while searching among huge amount of increasing information published on the web.

To achieve high efficiency in web searching, statistical method is proved to be an effective way and it can be interpreted from the semantic level. Based on the frequency of joint keywords, the keyword list can be generated and linked to each other to form a meaning structure. A data model is built when a proper keyword list is achieved and the model is applied to the design of the smart search engine.

Keywords: Search Engine, Google, Information Theory, Information Retrieval, Semantic Web

Acknowledgements

I would like to take this opportunity to express my sincere gratitude to my supervisor Dr. William Song for his advice and encouragement through the process of working on this project.

Moreover, I give special thanks and dedicate this thesis to my family. I thank them for their encouragement and support throughout my years of studies.

Table of contents

Figure list:

Chapter1 Introduction

1.1 Background

With a worldwide use of Internet providing people access to all the information resources from albums to online documents, it becomes difficult for users to find the exact information because the number of web pages on World Wide Web is growing increasingly with fast pace on a daily basis. Apart from the difficulty for computer systems to understand the exact meaning of human language input for search with complicated syntax, the consequences of returning inaccurate and unexpected results are common because of huge volume of data to be processed. Here is an example to show how a search engine works: a user enters “Search Engine” in Google, one of the world most popular search engines, and it returns 223,000,000 results. For the first few pages, none of them are related to how search engine works. Due to the complexity of human language, the computer can not understand and interpret users’ queries well and it is difficult to determine the information on a specific website effectively and efficiently because of the large amount of information. There are two main problems in this area: first, when people use natural language to search, the computer can not understand and interpret the query correctly and precisely. When a user types “Search Engine” in Google, Google doesn’t understand what the user actually needs, so it just simply uses a static pre-defined algorithm to compare the keywords “Search Engine” with the huge index file in the Google database, which stores the resulting index of words sorted by the indexer from every word on every page obtained by a web crawler (Blachman, N & Peek, J, 2003). Second, the large amount information makes it difficult to search effectively and efficiently. In 1998, Google indexed 26 million web pages and by the end of 2000, the number had jumped to one billion web pages. According to Claburn’s research in 2008, Google index reaches one trillion web URLs (Claburn, T, 2008). The Google database becomes one of the top ten databases in the world in terms of its size and efficiency. Although there are a huge number of web pages being indexed by Google, the information is not organized and structured properly, for example, when a user types a query which involves inference and reasoning over a complex data set, it is difficult for the current search engines to process the queries in natural language and return satisfied results from that huge database.

As a widespread solution to finding information on the Internet, search engine is commonly used in people’s daily life. Lots of companies provide search services on the Internet, such as Google, Yahoo, MSN and ASK.com. A search engine is a tool that helps people to efficiently find and retrieve information as requested from the Internet. There are two types of search engines: crawler-based search engines and human-powered search engines. A crawler-based search engine utilises automated software agents called web crawler or web robot to find and fetch web pages from websites, which are to be sorted by an indexer and stored in database. When a user gives a search query, the query processor will compare it to the index (file) using certain page rank algorithms and return the recommended documents which are most relevant to the topic (Blachman, N & Peek, J, 2003). In this project, these types of

search engines will be discussed and analysed because it has become the predominant search solutions nowadays. The second type of search engine mentioned above is human-powered search engine, where users submit information to a service providing company that indexes the information in different catalogues. These are also known as web directories and some links with high accuracy and quality are included in it.

In this thesis, the research work concentrates on the retrieval of text information from PDF documents and HTML documents. Compared to those research fields in image and voice search, the main research topic in this paper is to build up a semantic model from the collections of text information in order to help people easily find what they are looking for. For example, a semantic model can be extracted as the picture shown below:

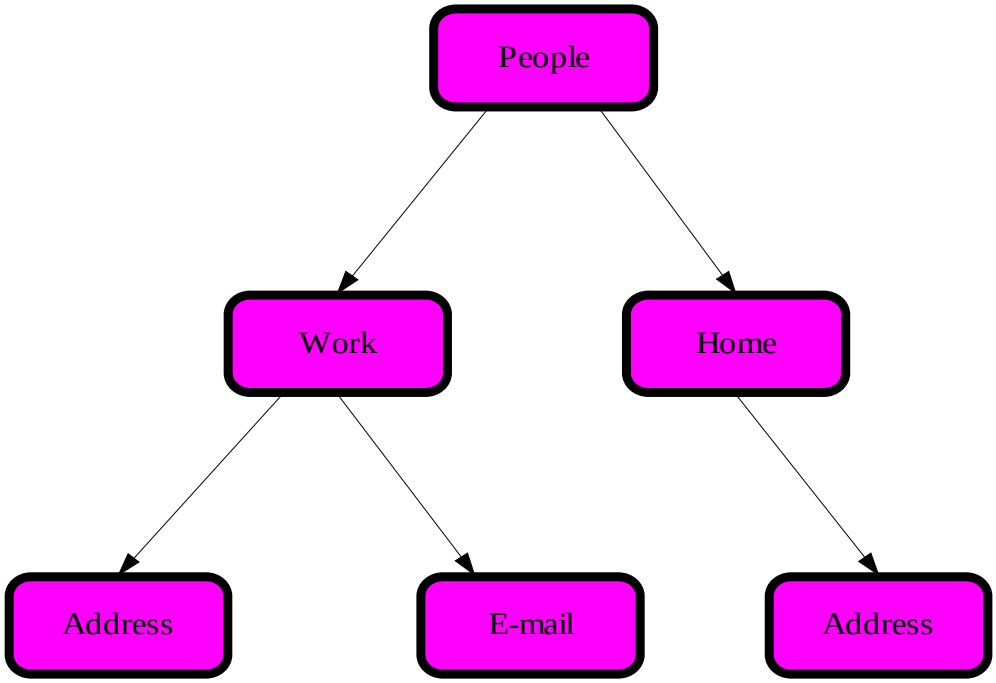


Figure 1.1: Semantic model

The semantic model, built from the resources, is used to assist users in their searching process to improve search efficiency and accuracy. In this chapter, we briefly describe the search engine technology, the concept of the Semantic Web and a combination of them, known as the smart search. Then we discuss the main problems of the current search engines and finally we propose a couple of possible solutions.

1.2 Search Engine

1.2.1 What is a Search Engine?

A Search Engine is a tool designed to assist people to search for information from the Internet. People may look for information that consists of web pages, images, music, movies and other types of files. According to the research and definition by Tharanga and Ranasinghe, “A search engine is a searchable database which collects information

on web pages from the Internet, and indexes the information and then stores the results in a huge database where it can be quickly searched. The search engine then provides an interface to search the database" (Tharanga, W. M & Ranasinghe, D, 2006).

The most commonly used search engines include Google, Yahoo, MSN and Ask .com. When people type any keyword in the search bar, the search engines will scan the content of the internet and return the information that might be of interest to the user. Because there are millions of websites, certain technology is needed to locate relevant websites, based on the title, keywords, and text information supplied by users.

1.2.2 How does a search engine work in general?

A search engine normally has three parts: a spider, an indexer and a piece of search engine software.

(1) Web crawling robot/spider

The web crawling robot, also known as 'Web Spider', is a program or automated script that browses the World Wide Web in a methodical, automated manner (Web crawler, Wikipedia, 2008). It can be used by a search engine to provide up-to-date information and create a copy of all the visited pages to be stored in an indexer that helps to index these pages to speed up the search process. Regarding the automated script, it can be used for automating maintenance tasks, for instance, checking links or validating HTML codes, or even gathering specific information from web sites, such as email addresses, phone numbers within special areas. It will start with its seeds which contain a list of URLs and then identify all the hyperlinks in the page and add to the seeds, which will be recursively visited according to some selection policies.

(2) Web Indexing

In order to support the information retrieval of efficiency and accuracy, the search engine indexer collects, parses, and stores data to optimize performance for the search query. Most of the popular search engines focus on the full-text indexing of online, natural language documents (Clarke & Cormack, 1995). Index design incorporates different concepts from linguistic, cognitive psychology, mathematics, informatics, physics and computer science (Search engine Index, Wikipedia, 2008).

After a web crawl robot obtains information, the search engine indexes the data according to the algorithm and stores the indexing file in a database. The next generation of search engines aims to develop more intelligence parsing and indexing technologies which could understand and satisfy people's requests in natural language. This will be discussed in Section 1.3 Semantic Web.

(3) Searching mechanism

The searching software part searches information from the indexer based on a search query and return the matched results. But the search queries can be unstructured and

often ambiguous they vary greatly from standard query languages (Web Search Query, 2008).

1.3 Semantic Web

The Semantic Web, as an evolving extension of World Wide Web (Semantic Web, Wikipedia, 2008), provides a way where information can be easily understood and organised by machine, while being available to define the data and semantics of information and services on the web. As Tim Berners-Lee described in his vision that the web will become capable of analyzing the data and in the future people's daily life will be handled by machines talking to machines. With the keyword based search engine, for instance, when a user inputs a query like "Best dinner in London", it will return the articles containing the keywords but no reasoning on the keywords regarding their relationships and page ranking. The first step to achieve this is to find a better way of semantically describing the web.

According to Gruber's research in 1993, the core part of semantics is the ontology. Ontology is an explicit and formal specification of a conceptualisation of a domain interest (Gruber, 1993), which consists of a finite list of terms and relationships between these terms (Antoniou & Harmelen, 2008). Davies, Studer and Warren define that an ontology consists of concepts (also known as classes), relations (properties), instances and axioms and hence a more succinct definition of an ontology is a 4-tuple (C, R, I, A), where C is a set of concepts, R is a set of relations, I is a set of instances, and A is a set of axioms (Davies, Studer & Warren, 2006).

Ontology can be described as a group of information, combined by concepts and relations. In order to express the Semantic Web, some formal specifications like Resource Description Framework (RDF), data interchanges formats, notations such as RDF Schema and Web Ontology Language (OWL) are developed to provide a formal description of concepts, terms and relationships within a given knowledge domain (Semantic Web, Wikipedia, 2008). In this heterogeneous information environment, a Semantic Web supported search engine aims to provide precise results to a customer query by describing the meaning of information more explicit with formal specifications tools.

1.4 The Smart Search (Combination of current search engine and the Semantic Web Technology)

1.4.1 Why a combination of the current search engine and the Semantic Web?

The Semantic Web provides a new solution to exploiting and managing information on the websites, which adopts metadata to describe the explicit meaning of documents,

not only from the aspect of the content of information but also from the relationships between them. While the information can be extracted from the natural language, it's a key issue to support a search engine to understand the exact semantic meaning of a word or text aggregation, both linguistically and logically. Ideally, the results returned by a search engine should not be a list of unrelated topics but a list of information properly organized, with low redundancy.

However, using these current well-known search engines, the results returned are not always satisfactory, and they include many useless records. Normally users have to read through several pages to get what they really want. These are keyword-based searches in which relationships between keywords are normally ignored by the search engines. This is quite obvious from the user's point of view. These are the constraints of the current search engines. In order to achieve lower duplicates, higher efficiency, more precise results, the Semantic Web technology is applied to process these keywords submitted, and the relationships between the entities some using extensible knowledge representation language known as RDF Schema and OWL. During the experiment development using the Semantic Web technology, we find that the current solutions cannot be simply replaced immediately because they lack the ability of processing relationships between keywords. A semantic search may discard all those web pages containing the keywords but with no semantic relationships, thus bring less search results. Therefore, a keyword based search engine and a semantic search can be integrated together to provide more options for search efficiency and effectiveness.

1.4.2 The Smart Search

The Smart Search proposed in this project, is an intelligent search engine that obtains information from websites, organizes the information using a semantic model or semantic description, and provides users with the organized results according to the meanings of the words and their relationships. It is a combination of the current search engine technologies and the Semantic Web based technologies, that provides more precise and human language based results. The smart search uses ontologies to describe the semantic model, which is self explanatory from complex queries, and searches the information from ontologies.

The objectives of the smart search are:

- Information on the websites should be reorganized according to the semantic meanings, apart from statistical data.
- Automatic and regular check by the web crawler for the latest web pages and information.
- With a user-friendly interface for the smart search engine.

1.4.3 An example to explain the smart search

From the users' point of view the smart search engine's user interface would be similar to most current search engines where the user query is submitted. For instance, when a user types "University" in the search bar, the smart search engine will return a possible collection of words that are related to the keyword, such as course

information, student gateway, lectures and research information. The exact keyword "University" may or may not appear in the results returned, which are actually related to the keyword and collected from the different web sites for the user to choose to narrow the search areas. The smart search engine will be able to interpret the natural language, study the meaning of it, and provide reasoning function through relationships represented in the ontologies.

1.5 Problems

The goal of this research is to build and test an enhanced search engine – the smart search engine – to help users to improve the efficiency and effectiveness of web searching. The new search engine is supposed to combine the current searching technologies with the Semantic Web technology by extracting and exploiting semantic model.

The main research questions addressed in this thesis include:

1. How to extract semantic model?
2. How to validate the semantic model that is extracted?
3. How to evaluate of the semantic model?
4. How to test the current semantic model extracted?
5. How to combine the current keyword based search engine and the smart search engine?

In order to find answers to the above questions, two experiments are designed and carried out.

1.6 Proposed Work

The Lucene search engine as an open source project, is chosen as the experimental platform to extract the semantic model. We start our experiments with a few selected words and other keywords as test samples to analyse the results of the experiments. With the fully tested semantic model, the principles and basic design of the smart search engine is produced for further development. In the following, the content of each chapter are briefly introduced.

Chapter 1 introduces the basic concepts and principles of the search engines, the Semantic Web and the proposed smart semantic search engine. Chapter 2 describes the major search engine technologies used in Google, Terrier and Lucene. Lucene platform is used to build our experiment platform. The Semantic Web and related information are also briefly discussed. Chapter 3 explains the fundamental semantic of the smart search theory and algorithms that are used for development of our search platform and experiment design. Chapter 4 describes the steps of the two experiments: how they are designed and conducted. Chapter 5 analyses the results of the experiments and possible solutions for the semantic smart search engine. The last chapter, Chapter 6, presents the conclusion and future research work.

Chapter2 Current Search Engine

2.1 Overview

The objective of this chapter is to demonstrate the existing research related to information retrieval, search engine, the Semantic Web, data mining. Some examples are provided to explain why and how these technologies are applied to solve the current problems and how to help us to build current semantic search engine. In this chapter, we try to address the relevant subject areas by analysing the current problems and remained problems with the current solutions. It will also discuss different subject areas that are related to this project.

2.1.1 Concept of information retrieval

Information retrieval, introduced by Lancaster in his book Information Retrieval Systems, is the act of identifying documents in the system. It is not just simple search for words and phrases, natural language processing and statistical analysis are involved to achieve the act. An information retrieval system does not inform (i.e. change the knowledge of) the user on the subject of his inquiry. It merely informs on the existence (or non-existence) and whereabouts of documents relating to his request (Lancaster, F. W, 1968).

Since problems appear when computers encounter high volume information, efficiency and productivity of searches concern most of people and basic knowledge of the terms and concepts are important to achieve these objectives. Here is a list of basic terms and concepts of information retrieval:

The document is a piece of text which a machine can “read” or “understand”, while terms describe the document and they appear in the document. The number of one term which appears in one document is called term frequency. The objective of current research efforts for information retrieval and data mining is not only to find documents with a given keyword or topic, but also to extract and integrate information among different documents.

2.1.2 Information Retrieval Systems

Information Retrieval System was initially developed in libraries (Staikos, 2000) and the first attempts at locating information of interest by the electronic means of fledge (Grossman & Frieder, 2004). Later development efforts are focused on probabilistic (Chaudhuri, Das, Hristidis, & Weikum, 2006) and statistical methods. When the World Wide Web was invented in the 1990s, to deal with heterogeneous and distributed data on web, design for the IR systems is evolved.

A typical information retrieval system structure is described by Van Rijsbergen as below:

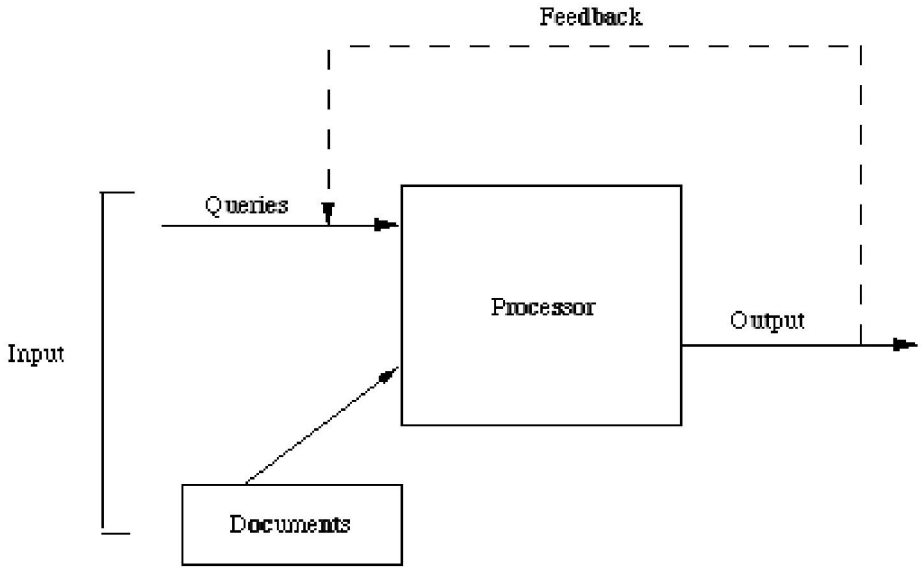


Figure 2.1 A typical information retrieval system structure

The process of information retrieval is described in Figure 2.1. The inputs, such as queries or documents, are provided for the processor to process. If the results as outputs are not satisfied, they will be returned as feedback to the current query, for a second time process. The main problem is to have these documents and queries well represented and readable for processor.

Meanwhile, the processor will be able to generate outputs as keywords extracted from all different inputs to make the documents to be searchable. Feedback is also provided to understand the queries better.

A typical Information Retrieval system can also be described as three main components as addressed by Baeza-Yates, Ribeiro-Neto, Salton and McGill in their modern information retrieval books:

- Documents are stored in a database associated with an indexer to generate representation for each document by extracting document contents.
- In the processor, there is a query system to determine the legitimacy of queries to search relevant documents
- A matching mechanism will evaluate how much the search results are relevant to the requirement submitted as query.

2.1.3 Problems of information retrieval system

Most Information Retrieval Systems (IRS) are used to store documents as information and use keywords to retrieve documents. By using different approaches, IRS extracts the keywords and then assigns weights to the keywords. The most concerned problem is about the precision of this process, that is, whether or not a keyword is extracted precisely and an appropriate weight is assigned to a keyword. Besides, as a daily update for the information in the document database, it is impossible to express the

information accurately and explicitly by using natural language and a traditional query from the users' point of view.

In general, an IRS tries to return users with most relevant results. The most popular search engines such as Yahoo and Google index billions of documents and normally take less than one second to return a complete answer list. The information required by a user would be updated very quickly and frequently, so it is even not included sometimes in the top result document list. Users have to read through the top ten or twenty returned results until they find the most recent information, which is ranked at a rather low position.

As a result, the users have to extract information by themselves by reading a long list of results to locate what they actually want. Naturally the primary problem areas that require attention for the IR systems are the ease of use compared to ineffective human computer interaction and the modern overabundance of information (Petratos, 2006).

2.2 Research on Current Search Engines

2.2.1 General overview

Popular search engines implemented as advance IR systems with the latest techniques and designs are extensively used by people, as introduced in chapter 1.2.1. Basically these search engines provide a (graphical) user interface to deal with the users' queries, retrieve and rank the results returned from the document database, where the web pages crawled by a robot are indexed and stored against the keywords the documents contain. In general the success of the search engines tends to be attributed to their ability to rank the results (Stopford, 2006). The ability to rank pages in the results which are processed and returned from several billions pages is a fundamental attribute for success.

Thus, reducing the time spent during the process of searching and improving the search efficiency becomes the ultimate goal for the search engines to achieve. In this section, we introduce and analyse some typical and popular search engines in relation to our research subject and objectives.

2.2.2. Google

In recent years, Google becomes one of the most popular web search engines that are able to process user queries and respond within seconds. Google was founded in 1998 by Larry Page and Sergey Brin and reached a net income of 3.07 billion dollars by 2006. Google's core technology is a page rank equation developed by the founders. The basic principle is that the system would include in the returned results a webpage with high ranking position, if this webpage is linked to many other pages. In the following section Google's architecture and core technology will be introduced together with its performance evaluation.

2.2.2.1 Architecture of Google

Like the other search engines, Google uses web crawlers to get various web pages and store these pages in the document database as a repository. These web pages will be parsed by the indexer as described in the figure below. After indexing, the content on the web are converted into a words list and these words are sent to the barrels, where information would be stored. These words are then sorted according to some principles, for instance, word occurrences (Brin & Page, 1998). When a user inputs a single query, the search engine firstly parses the query into words and stores the words into the barrels. Then the search engine scans through the web pages list and find the web pages that match all the words in the barrels. Finally, the search engine computes the rank of the matched web pages.

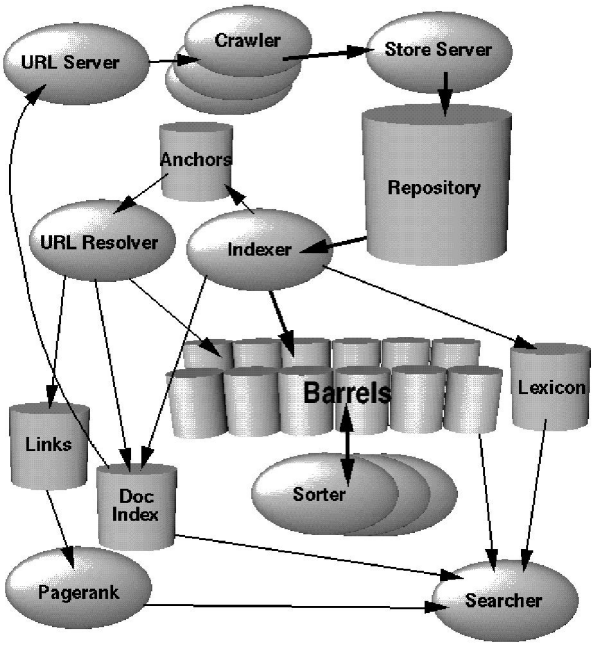


Figure 2.2 the architecture of Google (Brin & Page, 1998)

2.2.2.2 Page Rank Technology

The success of Google is attributed to the way it ranks web pages, i.e. the ranking algorithm to measure the importance of a web page. Quoting from the original Google paper, Page Rank is defined as (Brin & Page, 1998):
"We assume page A has pages $T_1...T_n$ which point to it (i.e., are citations). The parameter d is a damping factor which can be set between 0 and 1. We usually set d to 0.85. There are more details about d in the next section. Also C(A) is defined as the number of links going out of page A. The PageRank of a page A is given as follows:

$$PR(A) = (1-d) + d\ (PR(T_1)/C(T_1) + ... + PR(T_n)/C(T_n))$$

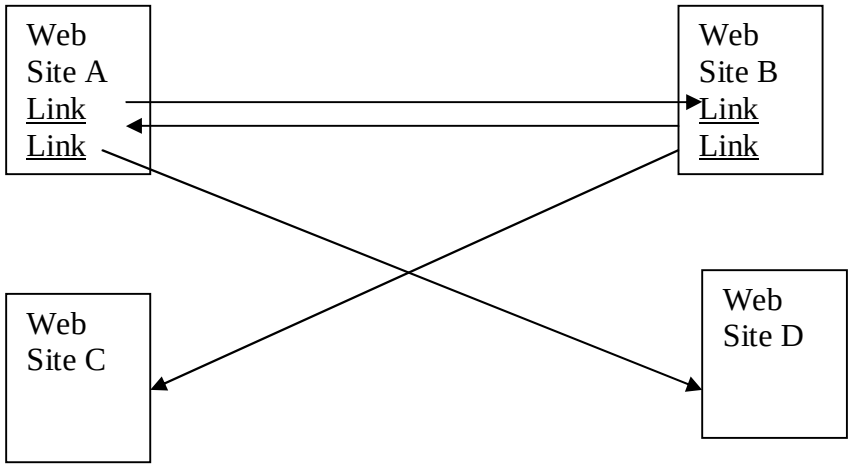


Figure2.3 The page rank of web site

Note that the PageRanks form a probability distribution over web pages, so the sum of all web pages' PageRanks will be one. PageRank or PR (A) can be calculated using a simple iterative algorithm, and corresponds to the principal eigenvector of the normalized link matrix of the web." It is determined entirely by the structure of the World Wide Web.

2.2.2.3 Evaluation of Google

To measure the quality of the Google search and users' experience with it, we focus on the attributes such as ranking efficiency and user interface availability. Since 2007, Google has launched over 450 improvements for search quality and evaluation. For instance, they make changes to handle links and anchor text, improve the segmentation of Chinese queries and new techniques that compound Swedish words, etc (Huffman, 2008).

Google has certain tenets to improve search quality, e.g., data driven in decision making. From understanding a user's requirements and intent, comparing search qualities with its main competitors, to improving the algorithm for search, Google employs a variety of evaluation methods and data sources, for instance, human evaluators and live traffic experiments.

Semantic technology used in semantic engine development is also employed by Google recently. That is, when a user inputs a single keyword for search, alternative keywords for the user to choose are provided at the bottom of the Google result pages based upon the initial keyword. This is a first clear example of Google employing a

semantic search engine that works by analyzing the context of words in their index and returning possible matches for meanings (Midwinter, 2007).

However, Google does not support natural language processing and it cannot be called as a fully fledged semantic search engine. Although Google can provide great amount of data and some users think it really helps them to understand the meaning of words and synonyms while the search engine itself cannot understand the natural language at the moment.

2.2.3. Lucene

Lucene is a highly sophisticating system yet simple to use open source for building up a text search engine project supported by the Apache group. It has a java based text search engine library that contains powerful indexing and querying functions. It was originally available as an open project on Source Forge developed by Doug Cutting in 1996 and in 2001 it joined Apache Software foundation's Jakarta family as open source server side java products. Because of the powerful functions, it is widely used by users who want to develop their own search engines, such as FedEx, Overture, Mayo Clinic, Epiphany and Hewlett Packard. Lucene supports simple text formats to which HTML, XML, Word, or even PDF files/documents can be converted and provides a tool development environment for writing the user's search engine.

Similar to the other search engines, Lucene consists of two components that make up a search engine, indexing and searching. Firstly, all the documents, originating from different text sources like web pages, or PDF files, are indexed in order to produce a common format. Preparing for a common format makes the search process convenient as the documents are processed by an Analyzer and turned into tokens to be actually indexed. Secondly, when a user inputs some attributes in the query, Lucene parses the query with Query Parser and creates search criteria which are used to run for the Query object against the index. Finally the items of data that meet the search criteria will be returned as Document objects to the user. This process is described in Figure 2.4 below.

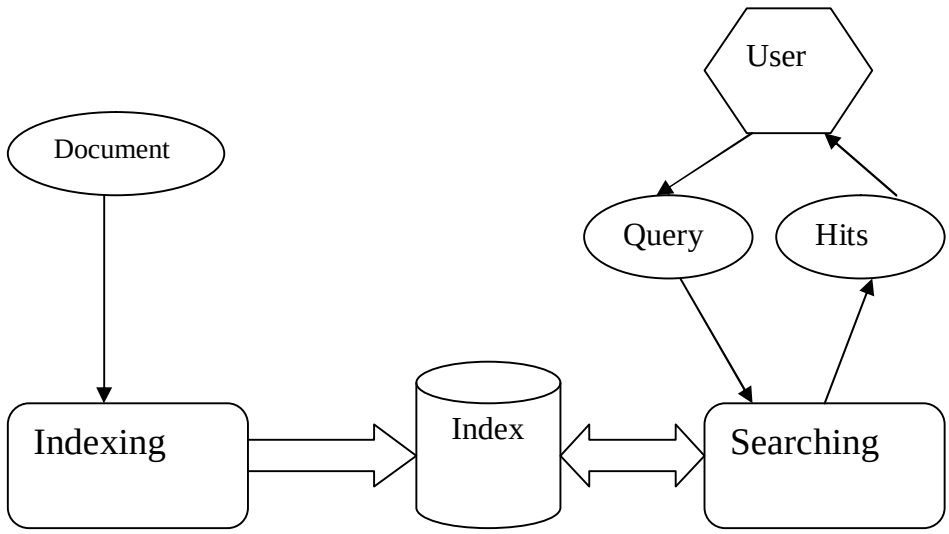


Figure 2.4 Lucene Text Based Search Engine

2.2.3.1 Lucene index process

With Lucene the indexing process is to convert all different file types into document objects with its name, title, category, simple description, written date, author's name and web links. These pieces of information will then be added to different sections and stored as keys in a Hash Map. The information can be retrieved from certain bucket based on the specific key. Unlike Hash map, Vector is used as an indexed list where terms and documents can be stored and expanded in a simple sequential data structure. Vector can be a mathematical method to measure the distance between terms and a COSINE similarity distance is used to determine how close terms and documents are. When a document object is created, the indexer will write this document to the index file. A standard analyzer is created to specify the directory where the index to be resided, whether a new index to be created or using the existing one, optimize and close the index etc. The index writer determines that an index file is created for the webpage.

The indexing process of the Lucene search engine is described as follows:

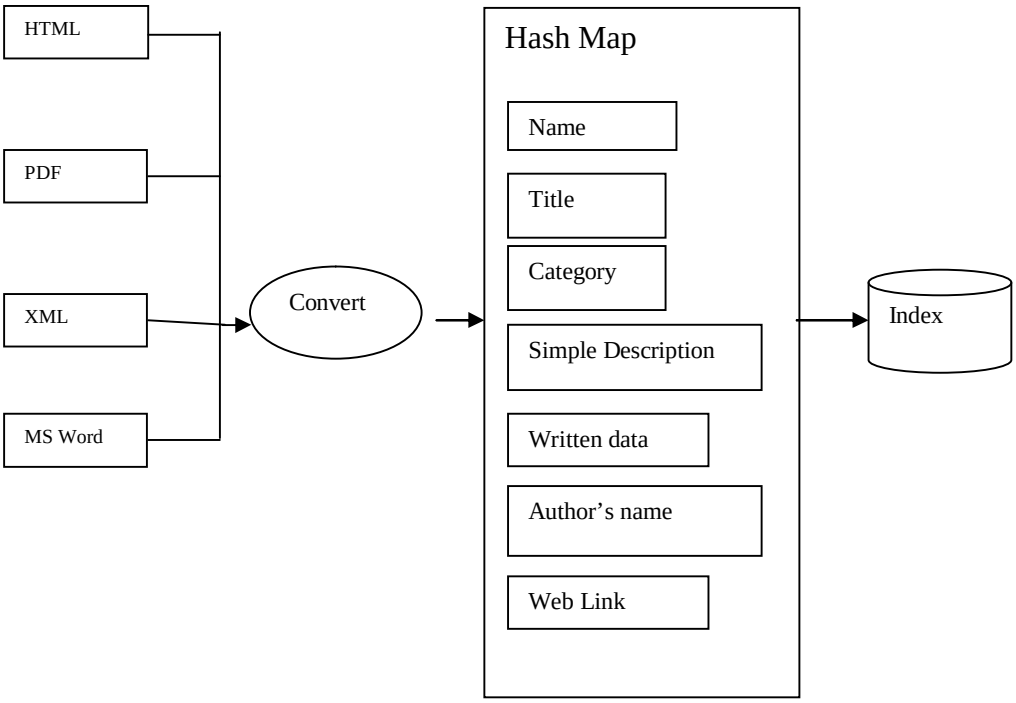


Figure 2.5 Indexing process of Lucene

- Conversion. The conversion process is to convert different types of file, such as PDF, HTML or Txt files to a stream of plain-text as tokens. Then Lucene can use these tokens to analyze the content of information.
- Analysis. The analysis function is to analyse data in order to ensure that it is suitable for indexing. For example, an analysis tool may remove the stop words that are the most common words but do not help searching, such as the words 'a' and "the". Lucene provides a stop word library which of course can be modified and updated according to the users' demands. The Indexer keeps

track of differences between words and all new documents to be added easily and merges the indexes as well periodically. The analysis tool splits text into tokens and performs operations for indexing process.

- Index Writing. When the process of analysis is done, Lucene adds all the information to the index which is used to store all the data. The tokens are extracted from the input document objects so as to generate keys for lookup.

2.2.3.2 Index formats and structure

The inverted index is a common structure used for many search engines. Lucene has two index structures (Cutting, 2006): multi file index and compound index. As Cutting described, the fundamental concepts with Lucene are index, document, field and term. An index contains a sequence of documents, a document contains a sequence of fields, a field contains a sequence of terms, and a term is simply a string.

Term is a token, the basic unit of indexing in Lucene, that represents a single string and word to be indexed after the document is transformed, including stop word elimination, stemming, filtering, term normalization and language translation (Goetz, 2000). The index also stores statistic data about terms in order to make the term based search more efficient and the indexes may be composed of many sub indexes also called segments, which can be searched separately.

One index consists of different segments and each of them is made up of several index files. Different index file share a same prefix but different suffix in a segment, for example here, we use SegName to describe the name of the segments and SegSize to describe the number of index files contained in that segment index. These index files can be identified by same prefix such as PrefixName, PrefixLength, but different suffix such as SuffixName, SuffixLength. The indexes will evolve by creating new segments for newly added documents and merging with existing segments (Cutting, 2006). So there is no need to index the whole content again each time when a new document is added. A simple example is shown in the figure below of the index structure of Durham University, which we use the Lucene index structure to index the university as a document.

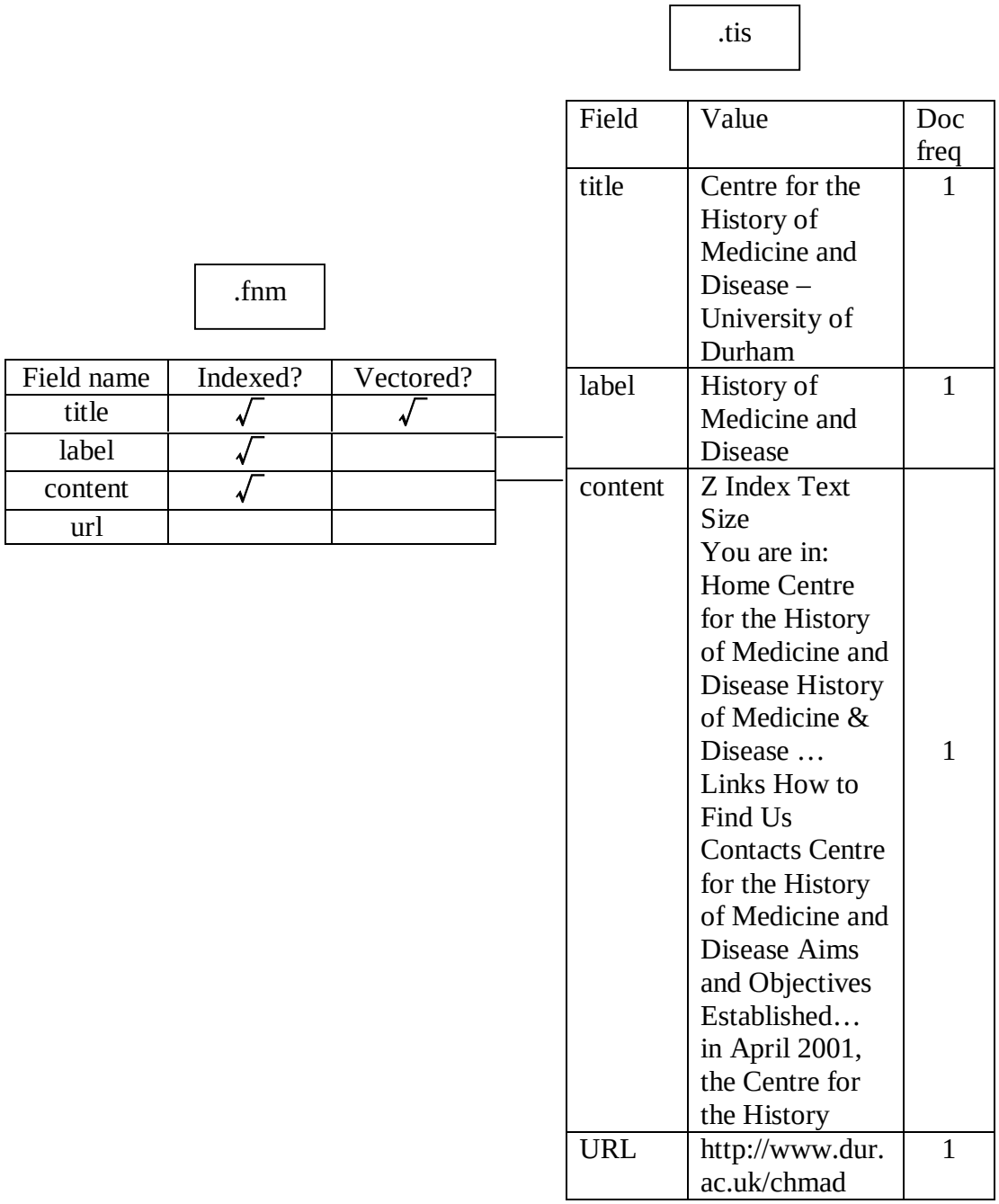


Figure 2.6 Index structure of Durham University

2.2.3.3 Lucene Searching

When an article is added as a document object and a user sends a search query from the front end interface, Lucene will run the query against the index to determine among the multiple search options. The following picture describes the process of the Lucene searching process. Firstly, the query will be parsed and converted to a list of

plain text, which is then processed by the standard analyzer of search engine, whose operations include discarding punctuation, removing accents, lowercasing, removing common words, stemming and lemmatization. The process is nearly the same as the Lucene indexing analyzer. Afterwards, the search engine searches the information from the different segments of the indexes and then returns the list of results.

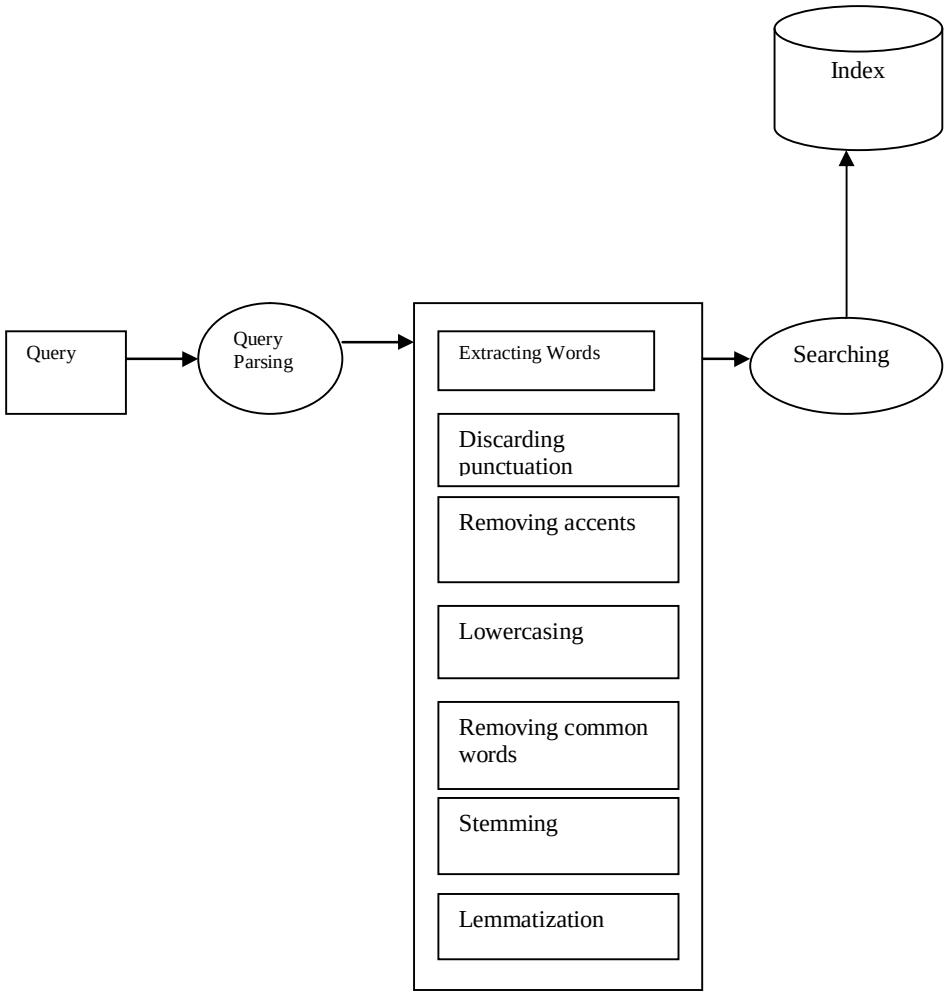


Figure 2.7 Index searching process of Lucene

2.2.3.4 Evaluation of Lucene

An easy way to get to know Lucene is to look at an example of using it. As shown in Figure 2.6 is an example of University of Durham, the Centre of the history of medicine and disease. Suppose that a Durham University professor publishes some articles and put them online for users/readers to be able to search. In contrast to Google, it could index the articles but could not process queries that require various search criteria, such as who wrote the article, which department the author works for, and any other topics published in the journal. Evaluation of Lucene can be taken quantitatively for question answering. It is also found that Lucene can find more documents containing relevant information. Lucene provides a good set of classes

designed to allow users' customization. Hence, an improved retrieval function based on Lucene can provide the search engine system with higher search accuracy. Using Lucene to develop a semantic search solution is possible as it can be modified with an advanced indexing component that is able to assign the words to nodes within the ontology.

2.2.4 Terrier

2.2.4.1 Overview of Terrier

Terrier, another highly flexible, efficient and effective search engine, developed in Glasgow University, is also an ideal platform for information retrieval based on large scale collections of documents. It has various features for indexing and retrieval functionalities including the parameter free probabilistic retrieval approaches, such as divergence from randomness models, automatic query expansion, reformulation methodologies and efficient data compression techniques (Ounis, Amati, Plachouras, He, Macdonald, & Lioma, 2006). The search engine is developed in Java and can be widely used and integrated into various information retrieval systems and the web search application developments can run on different operating systems and different types of computer hardware.

2.2.4.2 Indexing Structure and process of Terrier

Terrier is a transparent, easily extensible open source software project integrated with recent information retrieval methods and technologies. As an efficient test platform for large scale IR systems, it provides indexing and querying functions and offers four data structures including the direct index, the document index, the inverted index and the lexicon:

- The direct index stores the terms and term frequency for documents, including fields and block frequency.
- The document index is used to record document number, id and length.
- The inverted index stores the information like documents id of the matching document and the term frequency of that specific term in that same document.
- The lexicon: each term is unique with its id and statistics of that term. The information is stored in the lexicon.

A large scale of collections of documents are handled and parsed to form a stream of document objects, where a stream of terms is processed by the term pipelines as shown in the figure below. Eventually the information is written into the index data structures via the indexer. Extracted from each document, each term has three fundamental properties to describe itself: actual textual form of the information, positions where it occurs in the document, and the fields in which the term occurs (Ounis, Amati, Plachouras, He, Macdonald, & Lioma, 2006).

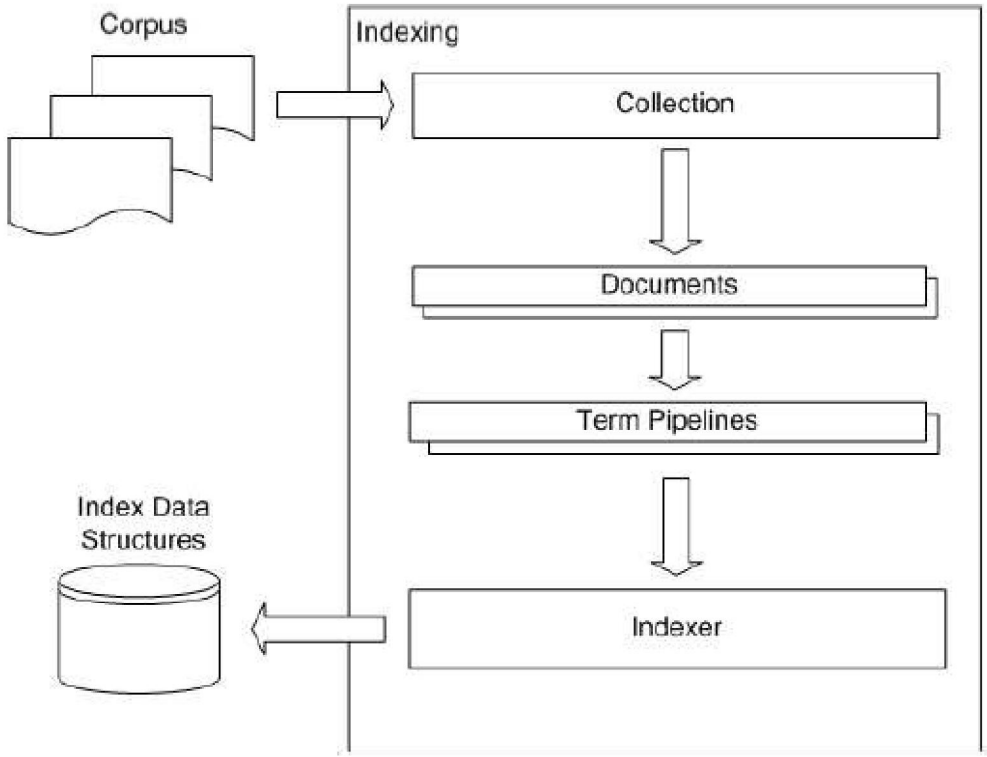


Figure 2.8 Indexing Process of Terrier

Figure 2.8 is an overview of the indexing architecture of Terrier. First the collection plug-in processes documents and generate a stream of objects as described above. Then the term pipeline transforms these objects and writes them to the indexer.

Apart from the indexing functions, Terrier is also designed to facilitate and assist research in Information Retrieval by having many retrieval features including weighting model, score altering for retrieved documents, advanced query language, and automatic query expansion (Ounis, Amati, Plachouras, He, Macdonald, & Lioma, 2006).

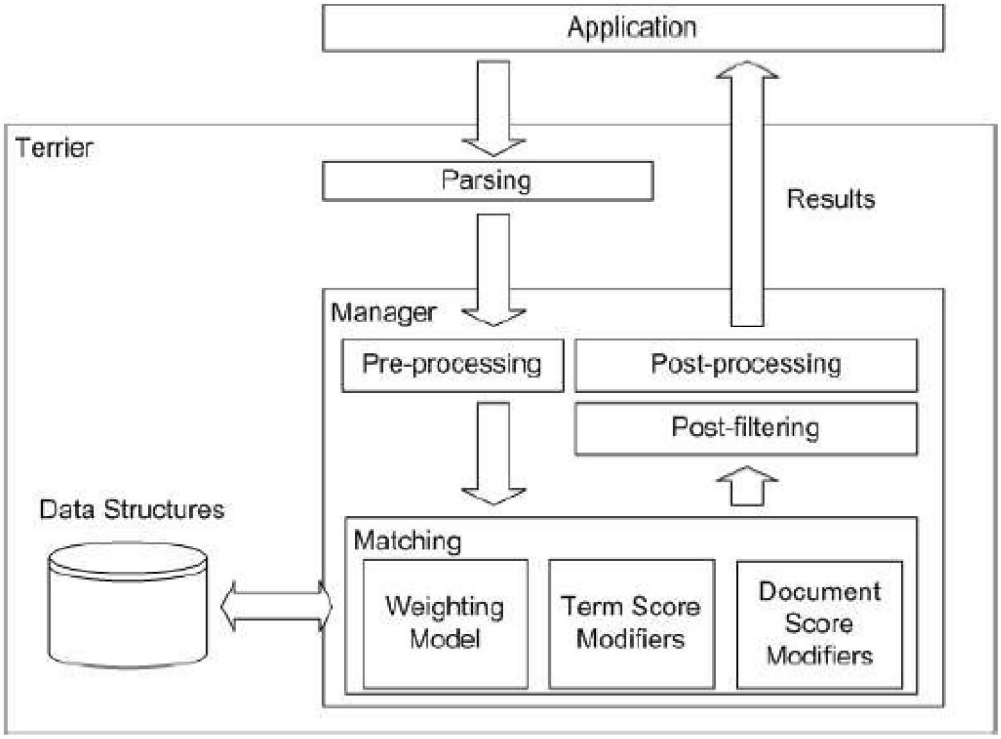


Figure 2.9 Retrieval Process of Terrier

Figure 2.9 is an overview of the retrieval process of Terrier. A keyword is parsed and then passed to the Manger Module, which will run the matching processing in turn and get the result after the post filtering. The weighting model will assign a score to each of the query terms in the document and help to identify the informative terms from the top ranked documents. The score can be updated for the individual term in a document or the retrieved documents by applying the term score modifiers or the document score modifiers.

2.2.4.3 Evaluation of Terrier

Terrier is designed for users to perform information retrieval research with a standard test collection and it meets the requirements for a rapid design and the state of art IR functions. It has evaluation tools to test the newly added methods and modules and for each module, an evaluation result against the test collections will be created and displayed once the test process is done.

Terrier can also index HTML documents, plain text documents, Microsoft Word, Excel and Power Point documents, PDF files because different document parsers are embedded. Its architecture allows any new parser to be added in to support additional file types. As mentioned above, for a new module, the test and evaluation process is quite straightforward.

Keyword based search engines are the main searching tools currently. However, some shortages still exist, for example, low precision, suitable keywords required for search, results sensitive to vocabulary.

Chapter3. Semantic Issues in Search Engines

3.1 The Semantic Web

3.1.1 Overview

The Semantic Web is an extension of the current web which was first introduced by Tim Berners-Lee, Hendler and Lassila in 2001. All the meanings of web information are all well defined and computer and people can work together for better sharing and reuse. Applying the Semantic Web paradigm as a universal medium for data exchange to search engines can have the good potential to bring significant benefits to scientific discovery.

Back in 1998, Berners-Lee described the concept of the Semantic Web as a plan for achieving a set of connected applications for data on the web in such a way as to form a consistently readable logical web of data-the actual implementation of the Semantic Web. As an effort promoted by the World Wide Web Consortium (W3C) to try to make the Web to be machine understandable, the Semantic Web allows users to transport any digital entity of information via this universal medium to any other systems and platforms. Thus, it improves human communications. The present web and all the information of these web pages are accumulated by human, and are not understandable for machine to process. When more information on the Semantic Web can be processed by machine, further semantic based processing such as reasoning becomes possible on the primary data from the web when described in a machine understandable form.

To identify some main problems related the current search engine, for instance:

- High recall, low precision: the keyword search engines not only return the relevant results but also return non-related results.
- Low or no recall: it means a user cannot get relevant answers from the search engine
- Results are highly sensitively to vocabulary: if the keyword a user chooses is not exactly the one (that they actually mean), the results cannot be found as expected even the word is related to the information.
- Results are single web pages: if the information a user needs is the combination of different entities, they have to first search each individual ones and merge them at last.

When the users get the result from the search engine, they still need to select the information and put it together according to their own understanding. This process is quite time consuming with people's effort involved and thus it is undoubtedly of low efficiency. The Semantic Web methodology proposes several standards for the current web so that the web information can be reorganised in certain format for machines to understand and interpret the meaning of information on the web.

Conceptual modelling can be considered as an effective approach, in which we can build a conceptual model for the web information and enable machines to understand and process the information from the web. Currently, the information on the web is presented mostly in HTML format. For instance, a simple HTML page that describes the staff information in a University would be something like this:

<h1>Staff</hl>
<h2>Lecturer John Smith<h2>
<p>Research student: Chris Golden<p>

A better, machine-understandable way to represent the above HTML information would look like this (in XML):

<University>
<Staff>
<Lecturer>John Smith</Lecturer>
<Research Student>Chris Golden</Research Student>
</Staff>
</University>

As can be seen here, most of the information can either be described in HTML or XML format which lacks proper structures and hence makes machines difficult to acquire and access information efficiently. Knowledge management, as one of the applications of the Semantic Web, aims to conceptually reorganise the information on the web and make it available and searchable for other machines and human.

3.1.2 Typical representation models

There are some typical representation models for the Semantic Web to meet the standards, such as OWL. Ontology, as defined by Studer and Warren, is an explicit and formal specification of a conceptualization. Generally, ontology is a group of terms and relationships between the terms. For the example described in Section3.1.1, the terms can be members of staff and his or her research students. The relationships between them can be a hierarchical structure of these terms (called classes in OWL). In this case, the relationship could be something like Mentorship between a mentor and a student. Additionally, ontology can include properties, specifications, restrictions, and statements. These properties can be very useful for web searching, data mining, etc. In the following, we will describe some main representation models here.

3.1.2.1 DOM (Document Object Model)

The document object model (DOM) is a tree-like model of objects contained in a document or web page. The DOM is precise and well defined to model HTML or XML sources. In order to capture more detailed semantic information, DOM can be used to define the logical structure of the document and web page and the way to access and manipulate them. Usually HTML or XML documents are the normal files to present various information stored on the web pages. DOM API can facilitate the process to build, navigate, add, modify or delete elements and content in these files. As a standard programming interface, DOM is designed to assist web structure analysis in different programming languages and a wide variety of environments. In its tree like logical structure, like the example in Section 2.3.1, the staff members and research students are the objects and the DOM model can encompasses its structure and the behaviour of the document and its objects. All the nodes like staff the members and students in the example represent the objects that have functions and identities.

As an object model, DOM identifies:

- The interfaces and objects used to represent and manipulate a document;
- The semantics of these interfaces and objects, including both their behaviours and attributes;
- The relationships and collaborations among these interfaces and objects.

3.1.2.2 RDF (Resource framework description)

Resource Description framework (RDF) is a W3C standard for describing resources on the web, for instance, title, author, content, copyright information, published date, etc. of a web page. In order to express semantics, RDF uses the structural constraints as a modelling method that enables encoding, exchange, and reuse of the metadata from the XML or HTML files. Thus, both human and machines can read and understand the RDF data modelling documents.

For any resource, the RDF data model uses a uniform resource identifier (URI) to identify the resource (RDF/XML Syntax Specification, 2004). Property types in RDF are used to express the relationships of values associated with different resources and the values would be either atomic in nature or other resources. Looking at the example in Section 3.1.1, we can use two statements to describe it:

1. “John Smith is a supervisor for research student Chris Golden.”
2. “The supervisor for research student Chris Golden is John Smith.”

To express the semantics in a machine readable manner, “John Smith” is replaced by a uniquely identified resource with the associate property types and the statement of the “The supervisor for research student Chris Golden is John Smith” has a single resource “Document 1”, a property type of supervisor and a corresponding value of “John smith”. The RDF model allows a higher order (multiple levels) logic expression to describe the inner- and super-structures of resources.

3.1.2.3 RDF Schema

RDF Schema, based on the RDF model, is used to declare vocabularies, a set of semantics property types defined for a particular community. RDF Schema is a primitive ontology language that can be used to describe the relationships between concepts for an application domain. Typical relationships between concepts defined in RDF Schema are subClassOf and subPropertyOf, which attempt to define one object class/property is a subset of the other. Simply speaking, RDF Schema defines a sub-concept relation between concepts, which allow us to reason about the concepts and the inheritance of attributes belonging to the concepts. In addition, RDF Schema also provides a few basic constraints on the concept/class definitions and property definitions. However, these constraints are very limited. As more and more ontology development is required for various purposes, a full-fledged ontology language – OWL (Web Ontology Language), has been introduced by W3C and is receiving increasing attention from the researchers.

3.2 Semantic Search Engine

3.2.1 Overview

Semantic Search Engines aim to improve online searching and generate more relevant results through reorganizing data from different web data models, providing semantic processor for search query parsing and conceptual structures for preparation of the web data when indexing them. Unlike the Google's page rank algorithm, a semantic search engine uses semantics to retrieve information from well structured data sources. As described in Section 3.1.2, some typical representation models provide formal languages to describe data with semantic meanings. DAML and OIL, as ontology languages, are developed to represent ontology. More tools like FaCT, Pellet, Racer and Jena are developed to create, edit and process OWL ontology which enables users to perform semantic searches based on reasoning.

Semantic search engines can be very powerful and complex because all the relationship information is stored in an ontology and the object or instance data are stored in a data file as part of the OWL/RDF descriptions. To deal with the problem, faced by the traditional search engines which have the limitation to search a large amount of heterogeneous datasets, the semantic search engines overcome the problem by setting up the semantic relationships among the datasets and reasoning the relationship information stored in ontology (Jeffrey & Hunter, 2006). In this section, some popular semantic based search engines are introduced together with their design principle and features.

3.2.2 ALVIS

ALVIS is an open source prototype of a peer to peer structured, semantic based search engine which was funded by the European Community. The system design is based

on using search objects as resources and automatically generates semantics to distributed queries and merge results. ALVIS does not rely on the existing Semantic Web ontology because it builds and maintains its own semantic structures and entities. ALVIS is designed to be able to operate with heterogeneous search servers, using query topics, distributed methods for ranking, and semantic based processing (Buntine, W. Valtonen, K. & Taylor, M, 2005).

3.2.2.1 ALVIS and Components

The major components of ALVIS are the input system, the document system, the maintenance system, the super peer and the P2P system as described in ALVIS super peer semantic search engine executive summary in 2005. The input system acquires documents from any sources (for instance, MS-word documents, HTML documents, PDF documents, etc.) then converts the documents into XML format. The functions of the document system are to prepare the XML documents for the runtime system and maintain the resources for the input system. The maintenance system is responsibility for processing document collection and semantic resources. The super peer component provides users with an interface and the P2P system is a collection of P2P sub-search engines and the P2P architecture handles distributed information retrieval by providing functionality for distributed indexing and query/retrieval as well as shared ranking (Ardo, 2005).

The overall architecture is represented in the figure below (ALVIS super peer semantic search engine executive summary, 2005)

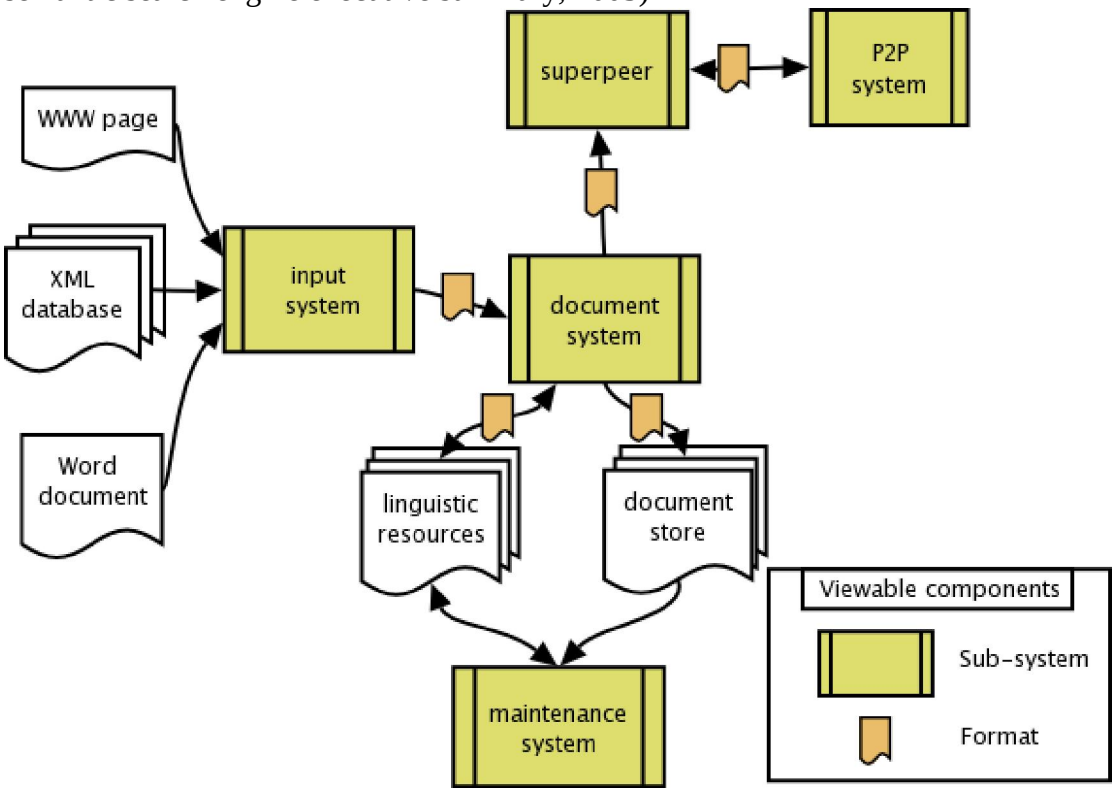


Figure 3.1 ALVIS architecture for search

3.2.2.2 Principles of ALVIS

ALVIS is not a traditional Semantic Web and it is built on the semantic technology which performs a semi automatic analysis on the web content. The input system takes in documents and converts them to XML files. Then the document system analyses the XML files semantically to build suitable indexes for the original documents. These results will then be available to the ALVIS network. Users can use the super peer to search information based on the ALVIS architecture.

3.2.3 Swoogle

Swoogle is another search engine for the Semantic Web. Ding and his colleagues (2004) described Swoogle as a crawler based indexing and retrieval system where it extracts metadata from all the documents collected by the crawler and computes the relations between the documents. Swoogle is designed for searching information from the Semantic Web documents (SWDS) and it is different from the traditional search engines as Swoogle is used for finding ontologies, finding data, and characterising the Semantic Web (Ding & Finin, 2004).

3.2.3.1 Architecture of Swoogle

Swoogle contains four major components in its architecture. They are SWD discovery, metadata creation, data analysis, and user interface (Ding, Finin, Joshi, Pan, Cost, Peng, Reddivari, Doshi & Sachs, 2004). The SWD discovery component is used to find the latest Semantic Web Documents collected by the web crawler and the metadata creation component generates metadata based on these Semantic Web documents. The data analysis component is to analyse the metadata and index these Semantic Web documents. The user interface is for users to submit queries and access searching results.

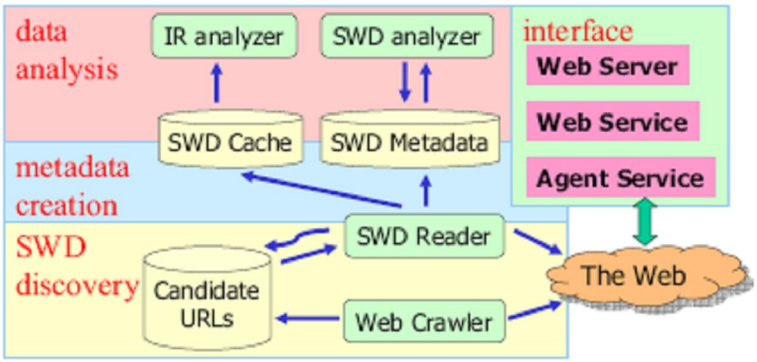


Figure 3.2 Architecture of Swoogle (Ding, 2004)

The feature of Swoogle is to support users to search Semantic Web documents based on the keywords as a query submitted via the user interface. All the results will be returned in a ranked order. The current version of Swoogle developed by Ding and his

colleagues in 2004 has discovered and analysed over 137,000 Semantic Web documents with 3,900,000 triples. More metadata on classes and properties will be captured. An ontology dictionary will assist Swoogle for powerful search and indexing functions.

Swoogle has the advantages of understanding the structure and semantics of Semantic Web documents, compared to traditional search engines. Since a plenty of online documents are written in Resource Description Framework (RDF) and online web ontology languages and become Semantic Web documents, Swoogle considers to customise its search engine to search ontology for the semantic annotation and reference, which can be used by web users, and even software systems (like agents) and web services.

3.3 Data Mining

3.3.1 Overview

Generally data mining is the process to analyse and collect data from different data source and summarise them into useful information. As described in Wikipedia, it is the process of extracting hidden patterns from data. As the amount of data and information have been rapidly increasing nowadays and selecting appropriate, relevant content is a very time consuming process, the data mining technique becomes an important method which can be used to uncover hidden information and messages. Basically, a piece of data mining software as an analytical tool can assist users to analyse data from different points of view for the data sets, including dimensions, categories, types, and subjects. The data mining process can also be regarded as a process of finding correlations or patterns among fields in large relational databases filled in by huge amount of data (Baloglu & Abdel-Badeeh, 2007).

In this section, some main data mining methods and techniques will be discussed together with its functionality and the way how it works. The project cycle of a data mining project will be introduced. Some typical data mining methods related to search engines for better performance will be investigated.

3.3.2 Data Mining and its Functions

Data mining is about analyzing data and finding hidden patterns with software tools providing automatic or semi-automatic ways to extract information from heterogeneous data sets. The data mining techniques can be widely used in different businesses including biomedical research, business analysis, forecasting, and customer relationship analysis.

The main purpose of data mining is to identify new patterns from datasets and predict the behaviour of the datasets and their interrelations, which may concern e.g. customers, products, and processes. The information extracted can increase its value upon the findings. For example, the data summarisation can provide tremendous economic value for businesses as for competitive advantages.

Unlike the traditional search engines or other querying tools that query with keywords and return relevant results, data mining aims to uncover the underlying facts that are previous unknown from the indexers or databases. Juneja and Phull (2007) consider that the data mining technology nowadays is done from flat files which have been extracted directly from operational data sources and it contains three stages:

1. First stage of search, with search architecture and evaluation of hypotheses,
2. Evaluation process of search output, and
3. Principle to use the results appropriately

Some applications of data mining are to identify and predict the opportunity and hazard. For instance, the software tools can be used for analysis of the best selling products, how to retain customers, which transaction are most likely to be fraudulent, what products and services the most customers are interested in, and what goods are commonly purchased, etc.

3.3.3 Data Mining project cycle

As a data mining project, there are a couple of stages and processes in the project cycle, which involves both human resources and data resources. Business understanding process, data understanding process, and objective and hypotheses definition process are considered (Collier, 1998) as the perspectives on data mining. Fayyad (1996) thinks selecting, transforming, evaluation as a unifying framework for a data mining and knowledge discovery project. According to Hofmann's thesis (2003), a generic data mining life cycle consists of sampling, data processing, data mining, modelling, deployment, and post processing.

Here we brief a simple process of a data mining cycle. When a data set is generated after gathering and sampling the data from the data collection process, irrelevant information is cleaned out from the data set. Based on the data types and values, the data are divided into a number of formats for the transformation process.

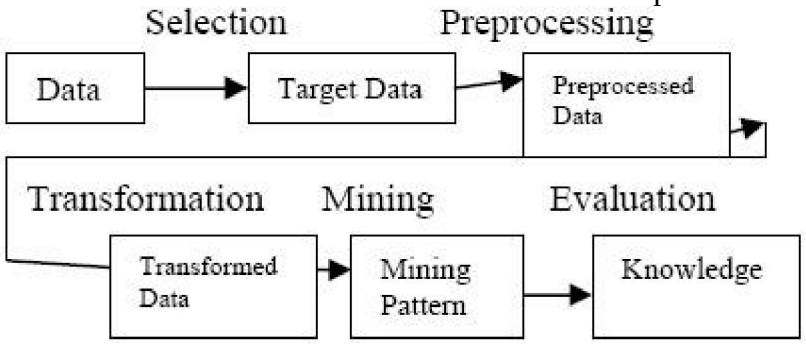


Figure 3.3 Data mining project life cycle

The modelling process uses algorithms and parameter settings to build the data model and patterns for assessment. If the standard for model assessment is met with the accuracy, the final evaluation process will take place to create a knowledge set, see the figure above.

3.3.4 Data Mining and Search Engines

Compared to the search engines in which users receive the wanted information from the indexers whose web data are collected by the web crawlers, the data mining tools analyse and extract underlying information from a database. A similar point for the two is that the data sets containing the raw data are very large and dynamic to change. It is a learning task for the data mining project and the final knowledge to be generated depends on the content of the current database. Nothing can be predicted if the relevant data sets don't exist in the data collection. When the information stored in the database is not entirely correct or accurate, it will cause uncertainty for the final results. Any changes to the current data sets, for instance, new information being added, modified or removed, will bring the learning system different results while the search engines will simply return more relevant results found in the indexers.

3.4 Summary

In this chapter, we discussed some existing research methods and approaches that are related to information retrieval and are useful to support the search engine design and development. We made a brief introduction to the current main search engines, such as Google, as well as some tools used as search engine platforms like Lucene and Terrier. Then we introduced the Semantic Web, together with its components and document formats in the related languages, and discussed how to use the Semantic Web technology to support and improve the search performance of the semantic search engine projects. We also investigated some well known semantic search engines such as ALVIS and Swoogle, focusing on their principles, architecture, and features. Finally, we discussed the data mining project, its functions, the way how it works, and the project life cycle in contrast to those by the search engines.

Chapter4. Smart Search Theory

4.1 The smart search theory introduction

In the theory proposed in the thesis, the smart search aims to determine a set of keywords for a web document, by which the semantic meaning of the document can be uniquely identified. Obviously, we expect that the size of the set of keywords is supposed to be small enough to be easily managed. This makes the fundamental assumption for the design of our semantic search engine. In this chapter, we will discuss the rationale of the assumption, the theory based on it, and the processes of how the theory is applied. We will also discuss the design of the smart search engine, in order to propose a solution to the efficiency problem when searching data from a huge yet increasing amount of information on the web.

In the analysis and evaluation of web searching, using statistical methods is proved to be an effective way, which can also be interpreted at the semantic level. Based on the occurrence frequency of the joint keywords (a keyword list), the key words in the keyword list can be semantically linked to one another to form a meaning structure.

Based on this idea, we build a data model for keyword lists from a sample web document space and apply the model to the design of the smart search engine. The idea contains three steps:
Step 1: To set up the process to find a keywords list;
Step 2: To identify and describe the relationships between keywords in the list;
Step 3: To apply the keyword list to build our smart search engine.

The three steps are an iterative process and through this process we can finally build up a conceptual document structure, together with a usable ontology, for an application domain, which will greatly improve the quality of semantic search.

4.2 Find a keywords list

4.2.1 Assumption

Again, the theory that supports the semantic search engine is based on the hypothesis that semantic meanings of a web document can be uniquely identified by a set of keywords of a manageable size and a set of relationships between these keywords.

Formally, suppose that R is a test space containing a collection of web documents and r_i is an element in the test space R, we have
$R = \{r^i \mid 1 \le i \le n\}$, (see Figure 4.1)

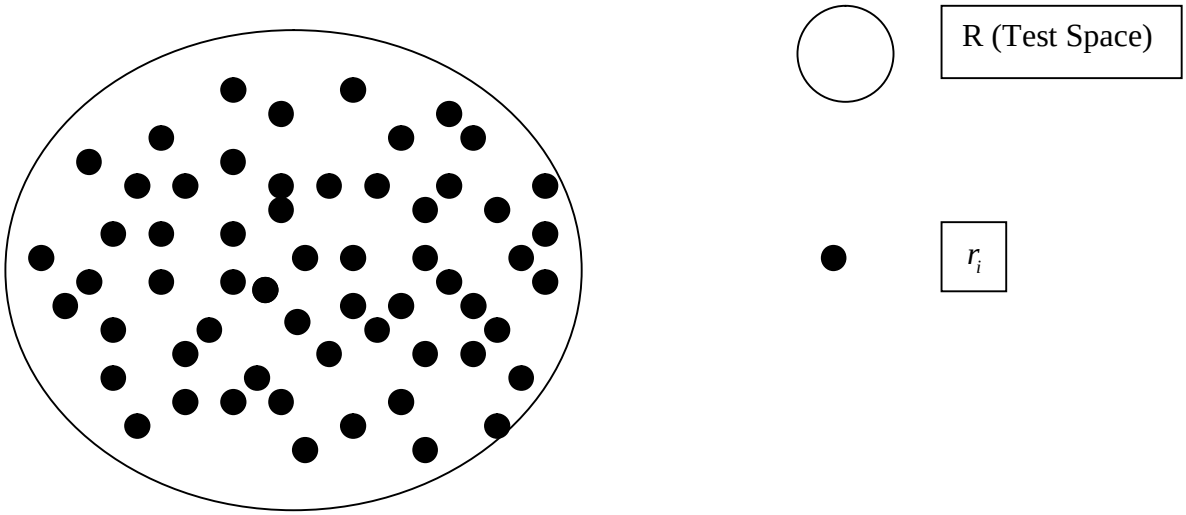


Figure 4.1 Relationship between R and r_i

Figure 4.2 below describes the relationships of a document A and its keyword list K_1, K_2, K_3,… K_n which can be considered as the parallel concepts/attributes.

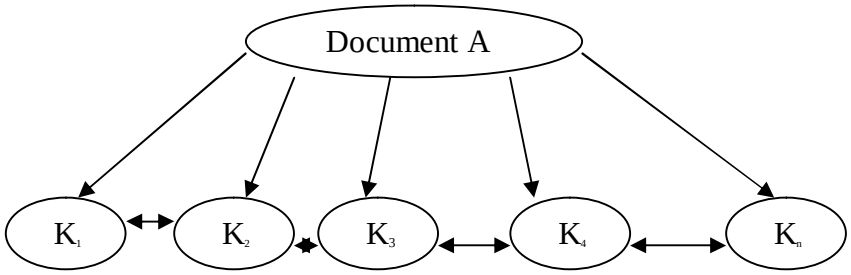


Figure 4.2 Relationships between A, K_1, K_2, K_3 …K_n.

We can also consider that A is a piece of information, such as a PDF document or a web page, to be searched.
Assume that K_1, K_2, K_3, …, and K_n are a list of attributes describing the information piece A. There are two simple relationships existing between A, and K_1, K_2, K_3, …, K_n.
(1) K_1, K_2, K_3, …, and K_n together place a restriction on A (i.e., the list of keywords defines the information piece A) or A contains K_1, K_2, K_3,…, and K_n. In other

words, the list of K = K_1, K_2, K_3, K_4,…, K_n can be used to describe the information piece A.

(2) K_1, K_2, K_3, … K_n are parallel concepts, that is to say that K_1, K_2, K_3, …, and K_n are not related to each other.

For example, there is a home page of a university lecturer, working at the department of computer science and having name Mark. The list of query keywords will appear as follows:

{"Mark", "department", "computer science", "lecturer"}

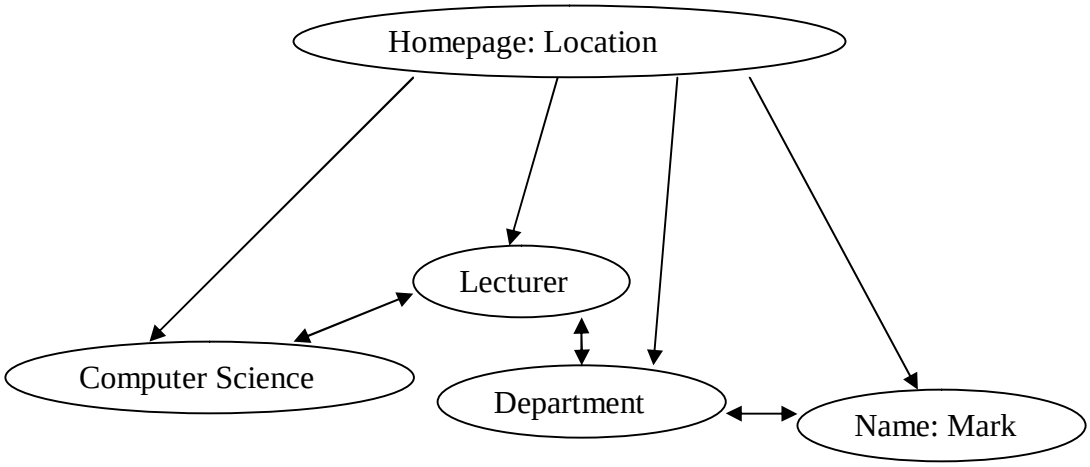


Figure 4.3 Example relationship

All the above attributes can be used to identify the homepage of the lecturer named Mark. Based on the above attributes, it is possible and perhaps easy to find the homepage of Mark and get further related information about Mark.

In the following, we make some assumptions as a basis for the development of the semantic search engine and we will also try to prove the concept of ideas in the assumptions.

Assumption 1 (basic assumption):

(1) For each document d, we have a word set k, so that k uniquely identifies d.

(2) For a set of document, d_1, …, d_n, we have a set of word sets, k_1, … k_n, corresponding to the set of documents, so that, for d_i, we have $k_i` = k_i - \{k_1, \ldots k_{i-1}, k_{i+1}, ..., k_n\}$ that can uniquely identifies d_i.

(3) Furthermore, for a word w` belonging to $k_i`$, w` can uniquely identifies d_i.

The assumption here is that any web document can be uniquely identified by using a set of keywords. At the worst situation to locate keywords, we need using the whole

text of a web article or document as a set of keywords to uniquely identify the document.

For example, to identify the web page of Prof. Simon Ross taken from the University of Durham website, we may use the whole text as the keywords to identify the web page, see Figure 4.4.

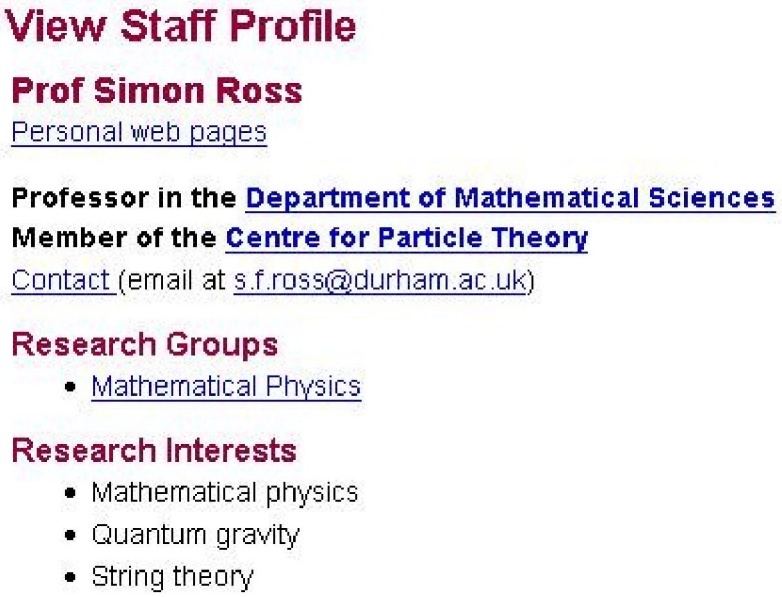
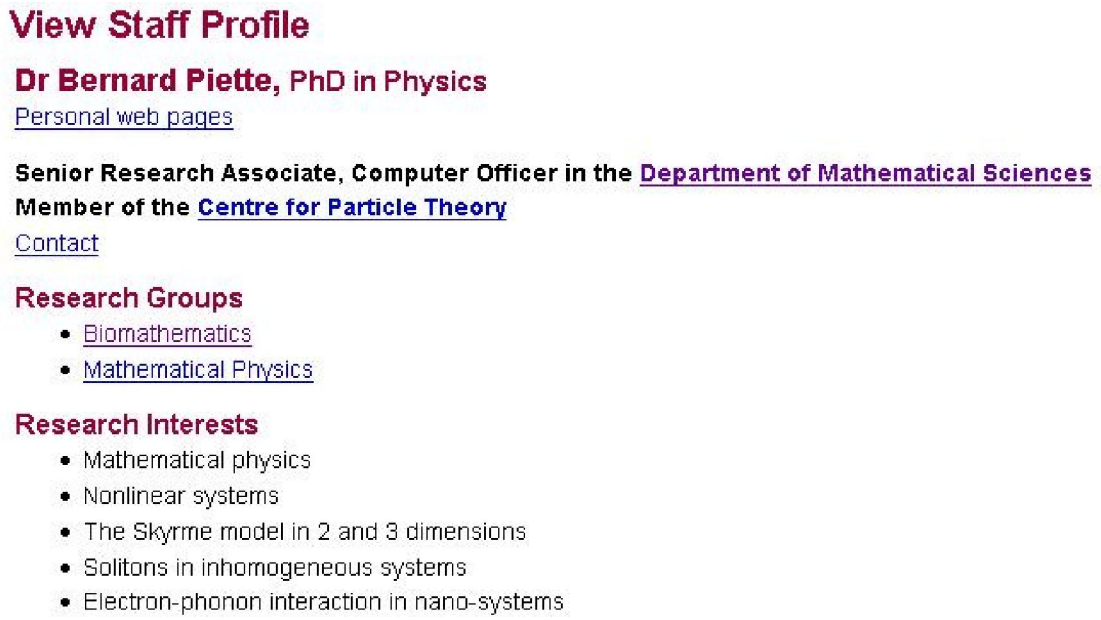


Figure 4.4 Prof Simon Ross in Durham University

In consequence, the keyword list {Prof, Simon, Ross, Personal, web, pages, department, mathematical, sciences, member, centre, particle, theory, contact, email, s.f.ross@durham.ac.uk, research, group, mathematical, physics, interests, mathematical, physics, Quantum, gravity, string, theory} can identify the webpage of professor Simon Ross in Durham University. According to Assumption 1, we can shrink this list to including only one keyword, e.g., the email address, which can still uniquely identify Ross web page given that the email address is unique at least within the university domain.

Following is another example of applying the assumption.


View Staff Profile

Dr Bernard Piette, PhD in Physics

Personal web pages

Senior Research Associate, Computer Officer in the Department of Mathematical Sciences
Member of the Centre for Particle Theory

Contact

Research Groups

- Biomathematics
- Mathematical Physics

Research Interests

- Mathematical physics
- Nonlinear systems
- The Skyrme model in 2 and 3 dimensions
- Solitons in inhomogeneous systems
- Electron-phonon interaction in nano-systems

Figure 4.5 Dr Benrnard Piette in Durham University

The keyword list {Dr, Bernard, Piette, PHD, physics, personal, web, pages, senior, research, associate, computer, officer, department, mathematical, sciences, member, centre, particle, theory, contact, research, groups, biomathematics, mathematical, physics, interests, nonlinear, system, skyrme, model, dimensions, solitons, inhomogeneous, electron-phonon, interaction, nano-systems} can uniquely identify the Dr Bernard Piette in Durham University.

For the second web document, all the texts are used as keywords to distinguish Dr. Bernard uniquely from Prof. Simon's web pages. To identify these two documents here, all the text of web documents above would have to be collected as the keywords to identify each single page.

4.2.2 Narrowing down the keywords list

Based on our assumption, the first step is try to narrow down the keywords list which identify the only web documents. For any given document d, this time we collect its title *t*, abstract *ab*, authors *au*, and document keywords k_w. Suppose that we build up a keyword list K which consists of the above mentioned items, i.e., $K = t + ab + au + k_W$ (for a html document, we use keyword, heading, etc. to make up K). For a set of document, $d_1,\ldots, d_n$, we have a set of collections, respectively $k_1, \ldots, k_n$, so that $k`_i = k_i - \{k_1, \ldots, k_{i-1}, k_{i+1}, \ldots, k_n\}$, that can uniquely identify d_i.

Note that we will build up a set of connotation relationships for these items that we picked for identifying the documents, which is briefly described as follows.

- Title – T: considering the contextual words for a given title t.

- Keyword – K_W: using a discipline category, e.g., Library Category, to describe the interrelations between the keywords.
- Abstract – Ab: applying a natural language parsing structure for building up the inner relations among the words in the abstract.
- Authors – Au: apart from the naming structure, we can also use a co-author relation.

For any two documents, d_1 and d_2, their identifying keyword sets k_1 and k_2, assume that $k = k_1 - k_2$, taking away all the duplicates we can say that k uniquely identifies d_1. Therefore, for a set of n documents, d_1, d_2, …., and d_n, and their corresponding keyword sets, k_1, k_2, …, and k_n, we can eventually find a set of keywords, $k` = k_1-\{k_i\}$, i = 1, 2, …, i-1, i+1, …, n, so that k' can uniquely identify d_1

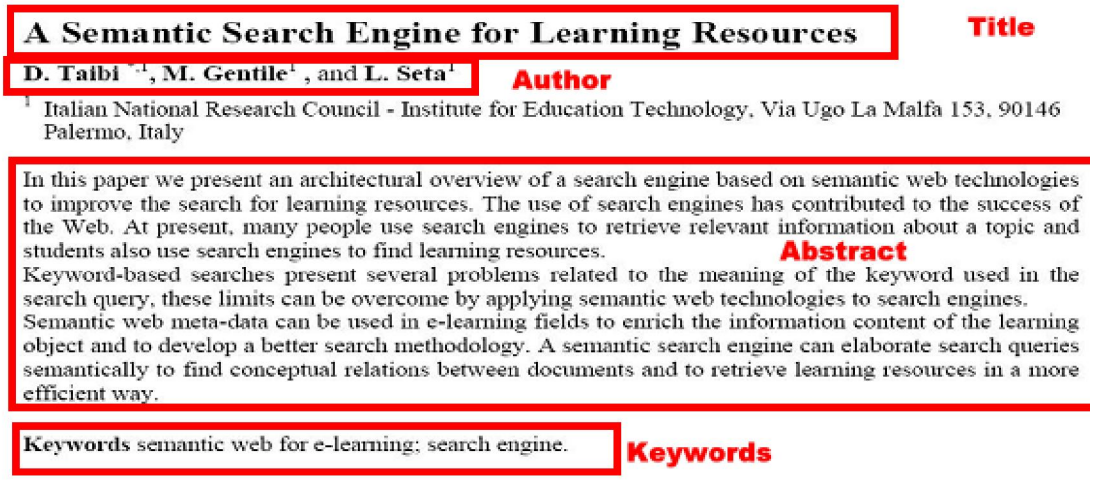


Figure 4.6 Web document, title, author, abstract and keywords

See the above example, Figure 4.6. To narrow down the keywords in an article, we take the title, abstract, author, keyword to analyse and all of these texts above can be used to identify this web document.

Following is another example to illustrate how we choose keyword sets for a HTML document.

```
<html>

<head>
<meta http-equiv="Content-Language" content="en-us">
<meta name="GENERATOR" content="Microsoft FrontPage 5.0">
<meta name="ProgId" content="FrontPage.Editor.Document">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Beergame Guide</title>
</head>

<body>

<center>
<h2>The Web Based Beer Game</h2>
</center>
<center>
<h3><i>Demonstrating the Value of Integrated Supply Chain Management</i></h3>
</center>
<h3> </h3>
<center>
<p><i><b>by <a href="contact.htm">Michael Li</a> and <a href="contact.htm">David Simchi-Levi</a></b></i></p>
</center>
<p> </p>
<p>Although the value of managing each component of the supply chain in order to
maximize customer service and profits is clear, the value of taking an
integrated approach to managing the entire supply chain is not so obvious. The
```

Figure 4.7 Source code of web document

In order to avoid using the whole texts of this web document as keywords, the content described in the sections, such as <Title></Title>, <h2></h2>, and <h3></h3> are used as a narrowed set of keywords to describe the document.

4.2.3 Narrow down the keywords list by using existing SHOES Ontology

The aim of building SHOES ontology is to determine and reason about conceptual relationships among concepts using the ontologic structure and the properties (also termed attributes) according to domain categories and their sub-categories, such as a library category. Let us take as an example the university-department ontology from where we hope to generate relationships and attributes for the web documents of the university.

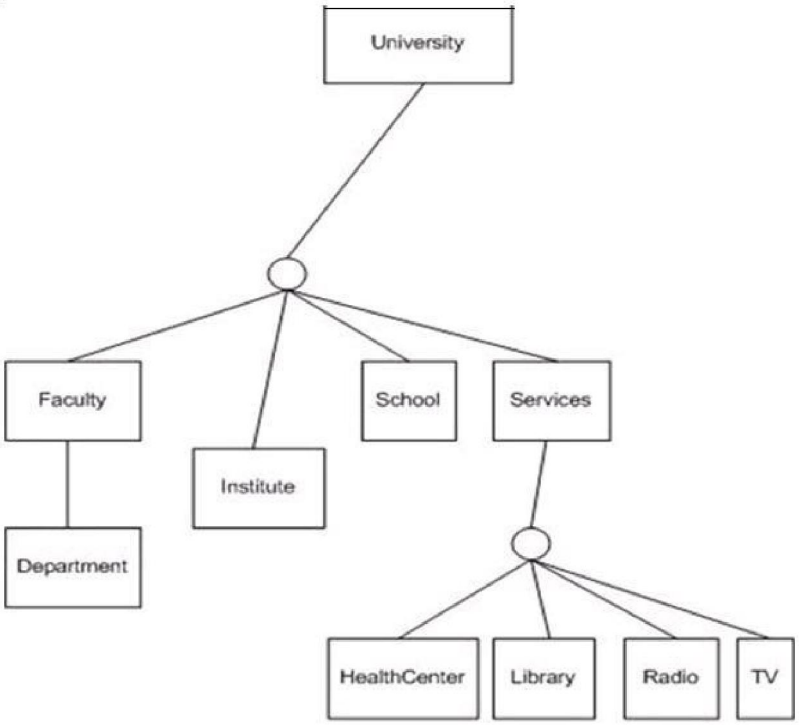


Figure 4.8 Ontology For Durham University

The university ontology illustrated above is a common model for universities. Normally, a university includes some categories under it, such as faculty, institute, school, and services. The sub-categories under a faculty include academic departments, and other services departments such as health care centre, library, and radio and TV station.

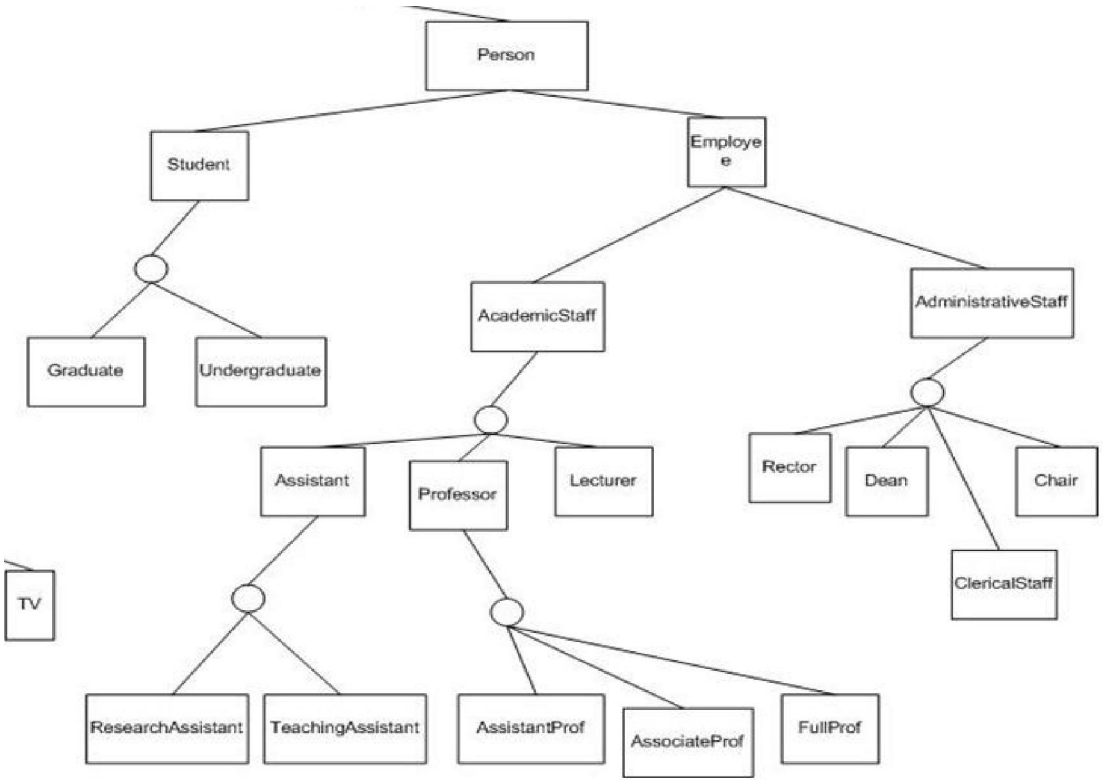


Figure 4.9 Person Ontology of Durham University

For the personnel categories within the university, there are students and employees. Undergraduate students and graduate students belong to the category of students. Academic staff and administrative staff belong to the category of university employees, see the figure above.

We use SHOE – Simple HTML Ontology Extension – for describing general university ontology. SHOE is a small extension to HTML, which allows the authors of web pages to annotate their web documents with machine-readable information (SHOE introduction, 2002). The existing SHOE ontologies are created to reflect the different context of modelled concepts, such as university ontology or person ontology, as described below:

The following taxonomy is the collection of categories declared in the graph above and the items with an asterisk means that they are associated with another ontology defined in the category, for instance:

Person*
Employee*
Academic staff *
Professor *
AssistantProfessor
AssociateProfessor
Lecturer
Assistant*
ResearchAssistant
TeachingAssistant
Administrative staff *
Rector
Dean
Chair
Clerical staff
Student*
Graduate
Undergradute
Postgraduate
University*
School
Department
Institute
Services*
Health Centre
Library
Radio
TV

In the following we define the relationships between different types or categories. The items with an asterisk mean the relationships have a local alias but are defined in other ontology, such as:

advisor(Undergradute, Professor)
affiliateOf(Univerisity, Person)*
affiliateOf(Univerisity, Undergradute)
alumnus(Univerisity, Person)*
DegreeFrom(Person, University)

This ontology below is for a PDF document and it has sub-categories such as Abstract, Author, Keyword and Title. There are relationships between this category with other categories. The items with an asterisk still mean that the category is also defined in other ontology.

PdfDocument
Abstract
Author
Keyword
Title
Author(PdfDocument, Person) *
AuthorOrg(PdfDocument, Unversity) *

4.2.3.1 Find keywords list using SHOES ontology

The key idea to use the SHOES ontology to find an attribute list AL from a list of keywords we prepare in advance for the test space for a category (called a term) in the ontology is: a) to search the term in the test space and obtain a number N_0 of documents which contain the term; b) to choose a keyword A_1 from the keyword list, search it together with the term in the test space, and obtain a second number N_1 of documents as return, and if N_1 is less than N_0 include A_1 as a attribute in A_L; c) to continue this process until a number N_x is found which is not less than N_{x-1}; d) then we consider that the attribute list A_L can determine the term. Next time when a user searches the terms in the ontology, we can provide her/him the attribute list for selection. We include the algorithm of this idea in our Smart Engine design.

In the following, we illustrate the idea in an example.

Here, the category from the ontology "department" is a term, i.e., Term ="department". The keyword list contains A_{1_1}= "chemistry", A_{2_2}= "news", and A_3= "university". A_1, A_2 and A_3will support to construct one of the attribute lists to determine the term "department". We are going to test whether they are attributes to be included in the attribute list.

The first step is to search the term "department" in the test space, see Figure 4.10. The number of result documents returned is 133,000, see Figure 4.11 (note: we used Apache Lucene to search in the sample space of Durham University websites).

Figure 4.10 Search for "department"

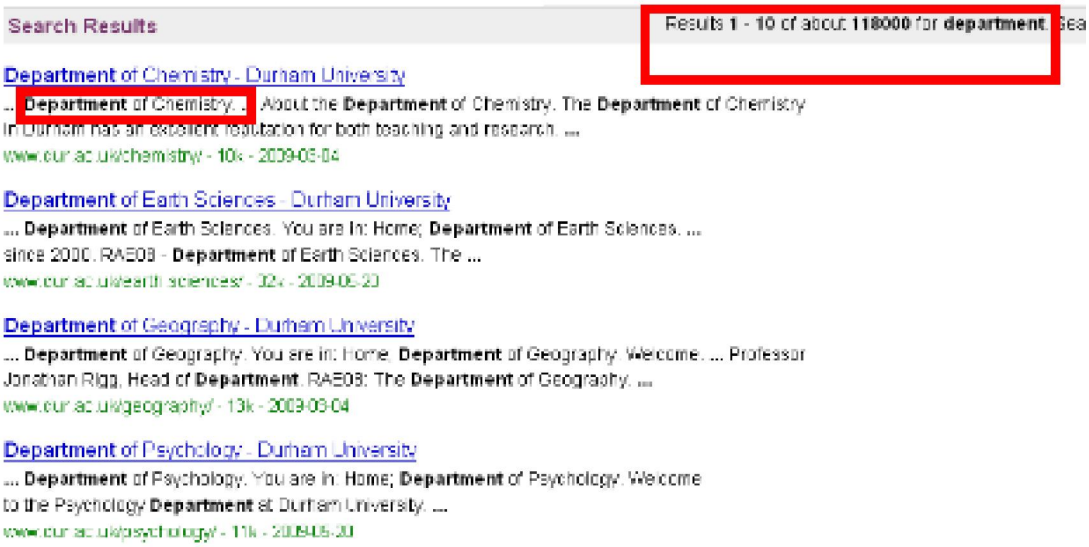


Figure 4.11 Results returned for “department”

For the second step we start to find the attribute list for the term “department”. As described earlier, we choose a keyword A_1, close to the term “department” from the results of the first search. Then we search A in the test space. If the amount of returned results is getting smaller, it proves that the keyword A_1 is one of the attributes for the term “department” and vice versa.

Here the returned result is 7,290 when searching the keyword “chemistry”, i.e. A1, together with the term “department”, see Figure 4.12 and Figure 4.13. So we include the keyword A_1 in the attribute list.

Figure 4.12 Search for “department chemistry”

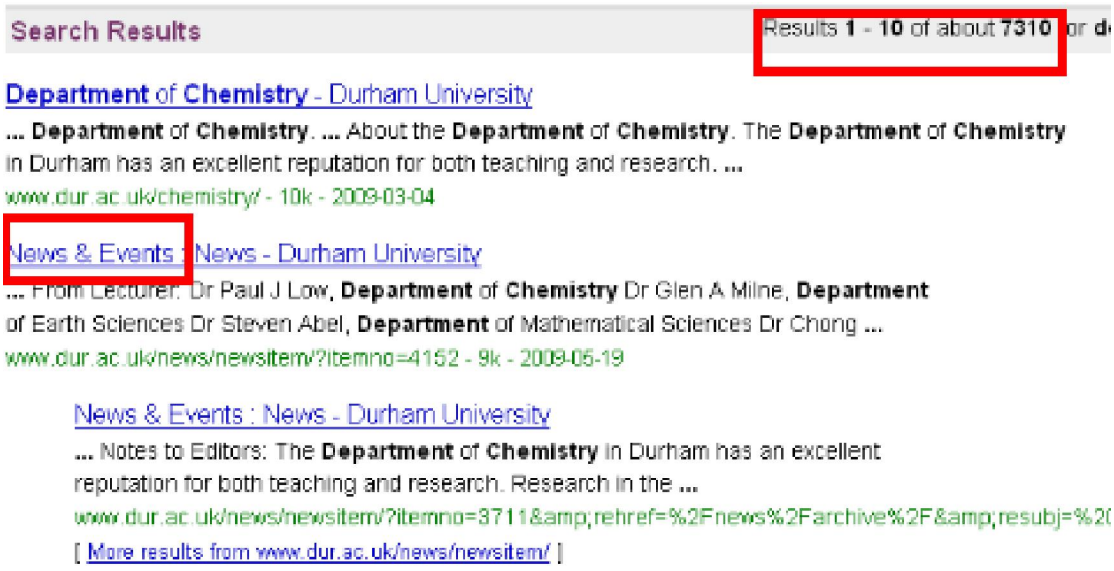


Figure 4.13 Results returned for “department chemistry”

For the third round, we choose the keyword “news” as A_2 and search it together with the term “department” and the attribute A_1 “chemistry” against the test space and the returned result is 1,280, which shows that A_2, “news”, is also one of the attributes of the term “department” and should be included in the attribute list, see Figure 4.14 and Figure 4.15.

Figure 4.14 Search for “department chemistry news”

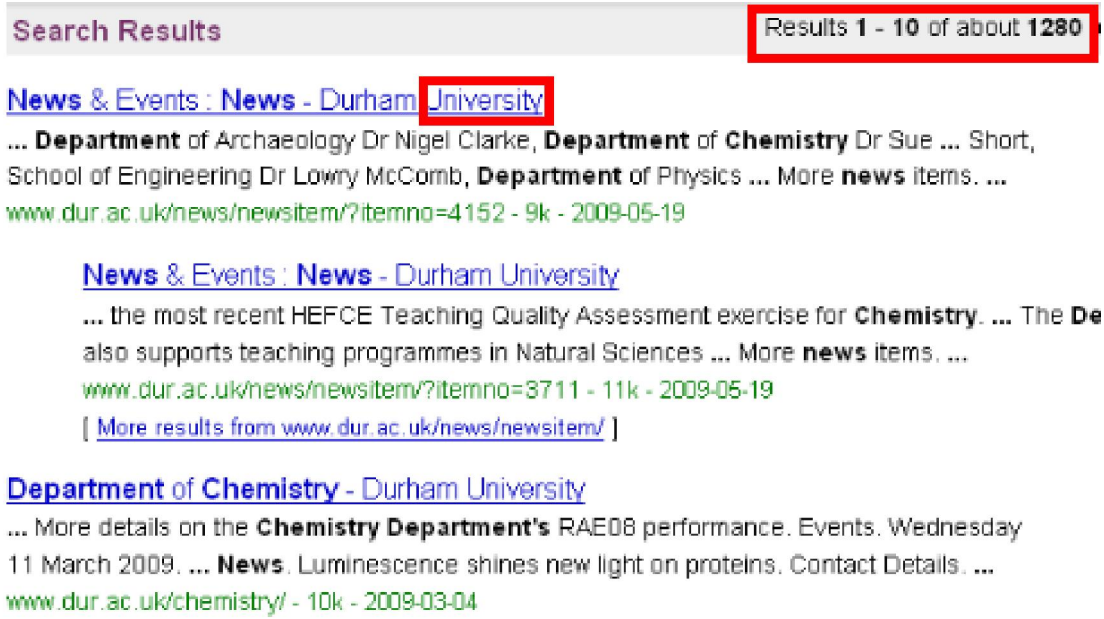


Figure 4.15 Results returned for “department chemistry news”

Now for the next round of attribute selection, we chose the keyword “university” as A_3 and search it together with the term “department”, A_1 “chemistry”, and A_2 “news” against the test space and the returned result is the same, with the number being 1,280, see Figure 4.16 and Figure 4.17. This shows that the newly chosen word A_3 has no influence to the results and so it should not be included in the attribute list for the term “department”.

Figure 4.16 Search for “department chemistry news university”

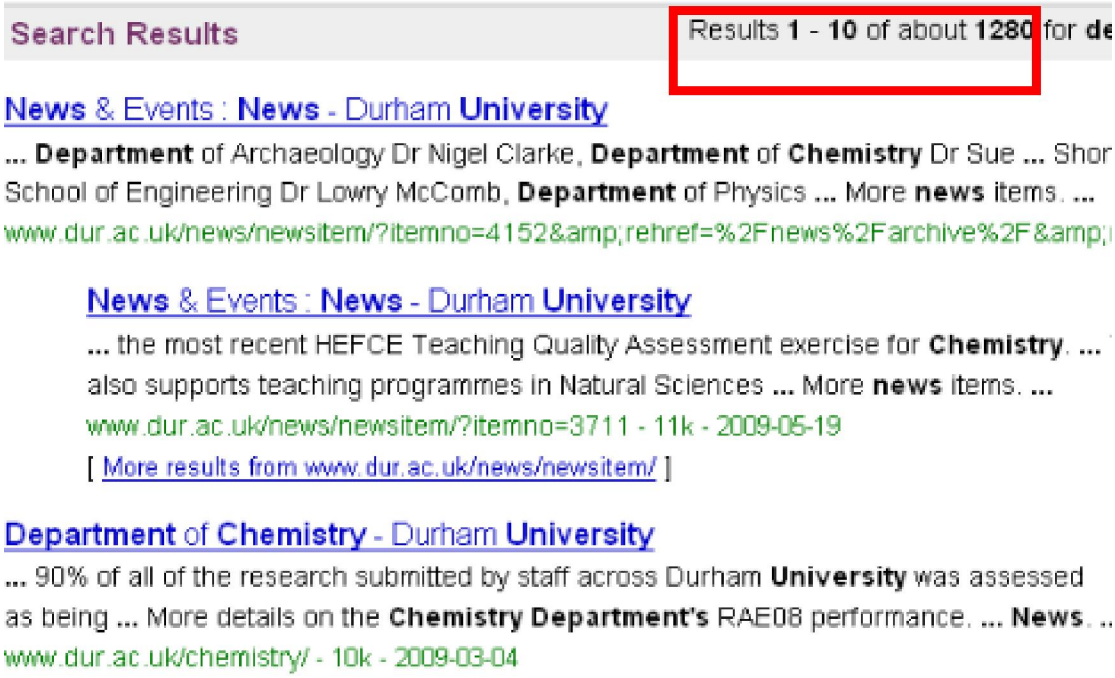


Figure 4.17 Results returned for “department chemistry news university”

In this case, both attributes A_1 = “chemistry” and A_2 = “news” belong to the attribute list used to determine the term “department” (note that it is a category in the SHOES ontology). Similarly, other attribute lists can also be achieved in this way for this term.

Then these attributes A_1, A_2…A_n will be stored in the database and when users search a term, the system will look up the words in the attribute lists and provide them to the users, and thus to help the users to refine their search criteria and improve the quality of search results.

For example, when a user types in the word ‘department’, the attributes ‘chemistry’ and ‘news’ will be selected from the attribute lists and displayed to the user for selection. The user can then choose any of the attributes to narrow down their search results.

4.3 Use the experimental results to build smart search engine

The main issue here is how to find and use information efficiently as most information from the web is organised in a weak structure or with no structure at all. The efficient use of such information encounters a big challenge, with the problems existing in the following areas:

- Searching information
- Extracting information
- Maintaining information
- Uncovering information
- Viewing information

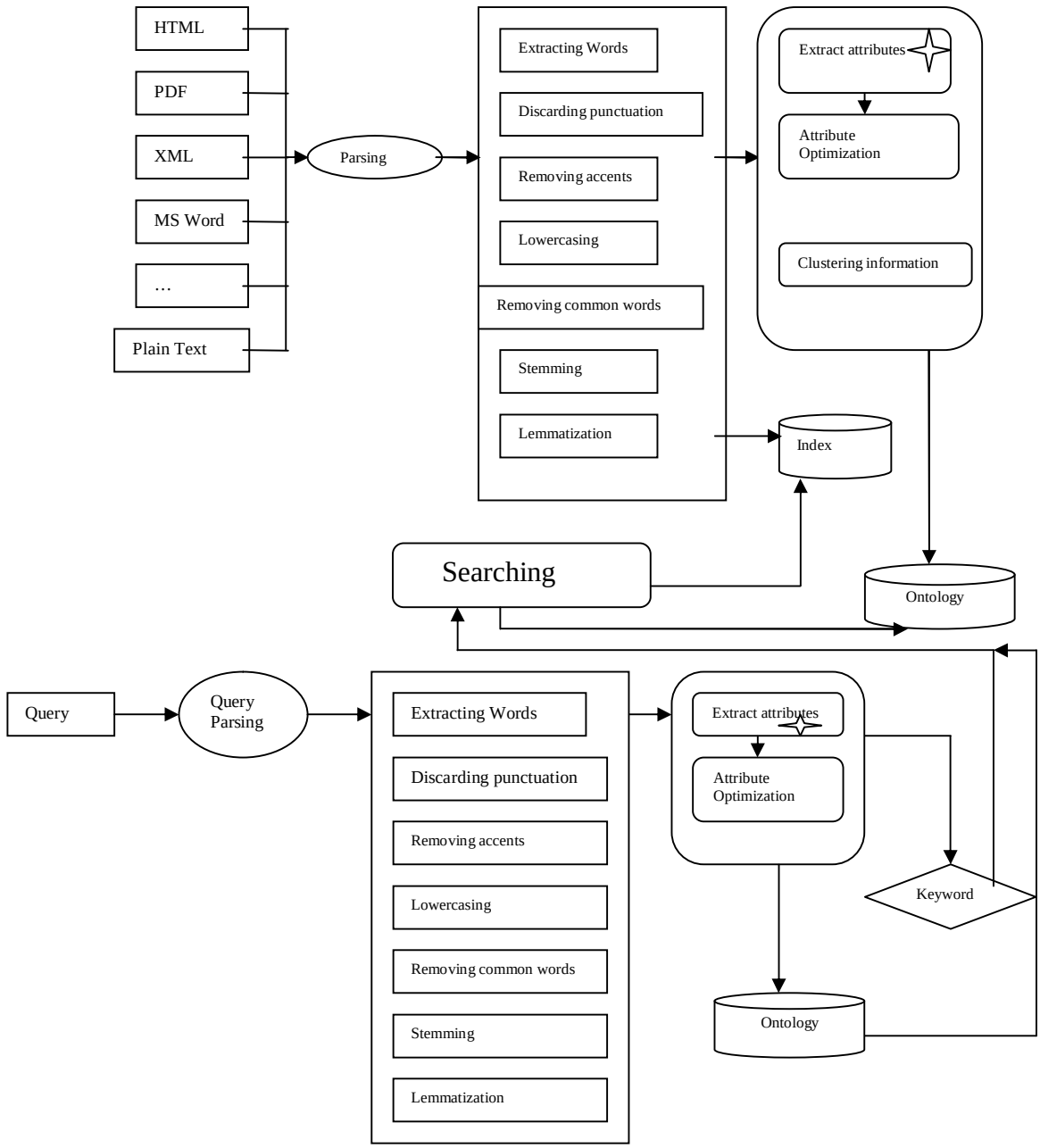


Figure 4.18 Structure of the smart search Engine

In Figure 4.18 the structure of the smart search engine is illustrated and the process of acquiring information and retrieval consists of the following steps.

- Firstly, different document formats are parsed in order to convert them into a stream of plain-text tokens.
- Secondly, the Simple Index Analyser (SIA) tokenizes text by performing operations, including extracting words, discarding punctuation, removing accents from characters, lowercasing, removing common words, reducing words to a root form (stemming), or changing words into the basic form (lemmatisation), which is similar to Apache Lucene.
- Thirdly, the Depth Index Analyzer (DIA) is designed to analyze the tokenized text in depth. According to the formula for different document formats (Different formulas were given to different documents, for instance, a web page like HTML documents or a paper such as PDF documents and attributes are to be extracted and optimized from the tokenized text. Attribute optimization means that information will be merged repeatedly in order to reduce redundancy. At the same time, two frameworks are used to support these functions. Clustering information is an important part of DIA, as it is useful for improving the efficiency of the search engine.
- Fourthly, for the Ontology building process, the optimized and tokenized text is used to build ontology that supports the searching process.

The steps of retrieving processes are concluded as follows:

1. Firstly, a query will be parsed to a group of words.
2. Secondly, the Simple Query Analyzer (SQA) performs the same operations on the query as described above in the step two.
3. Thirdly, the Depth Query Analyzer extracts attributes from the query and optimizes the attributes.
4. Fourthly, the attributes are used to build ontology (this will be the step for the ontology based search engine).
5. At last, the searching process will start from here.

There are a couple of reasons and advantages to promote the new intelligent search engine. Firstly, information will be reorganised according to its meaning and automated tools will extract more new information and maintain this information. Secondly, more friendly answers will be given according to the new query system. Questions can be input and the system will give users instructions which they can follow to attain the information whatever they want. Thirdly, users can get answers from different documents.

4.4 Summary

In summary, the smart search theory and its rationale are introduced in this chapter. Any web documents can be described by a set of keyword lists and the size of it can be minimized to a certain level. Finding the keyword lists is the key for smart search theory. It is an effective way to use statistical methods for keywords to analyse and evaluate web searching. From semantic level, the relationships of keywords in the list are also identified in this chapter, using both statistical methods and SHOES ontology. Finally, the data model is achieved to apply to the design of smart search engine.

Chapter5 Experiments

5.1 Experiments Overall

In order to perform an initial validation to the concepts described in previous chapters, the framework is built as experimental platform, for evaluation purpose. An indexer and web searcher (Apache Lucene) are available to process a small collection of 20,000 web pages in Durham University.

The following two experiments are based on the smart search theory and the hypothesis addressed in chapter four. Some main steps to carry out for experiment are briefly described here:

Step1: Extraction of keyword lists
The first experiment will first index the web pages of Durham University and then build the Lucene Search Engine with the 20,000 indexed data. Keywords will be extracted based on the SHOES ontology. The second experiment will extract keywords from the analysis of title, author and abstract information of PDF documents, collected from Google Search Engine.
Step2: Analyze keywords list and have them described in data model. Check the influence on the final search results when keyword list is changing.
Step3: Apply the data model to the smart search Engine design and testing.

5.2 Experiment one

There are many methods and suggestions proposed to improve the efficiency of search (Nakov & Bekiarov, 2008, Moura, 2005), in order to catch up with the increasing speed of information boom on the web. Most of these proposals are concentrated on term frequency and page rank algorithms and yet very few of them focus on the semantic relationship of the content. The objective of this research project is to provide a semantic relationship model based on the experiments carried out in this project. The model can be used for semi-structured or unstructured information on the web to help improve the accuracy and efficiency of search engine.
The semantic structures concluded from our experiments here can be applied to any web documents that can be identified by a minimum set of keywords. The purpose of the experiment is to extract the semantic model and evaluate the appropriateness of the semantic relationship model extracted from web page. The website of University of Durham is used as sample space and the Apache Lucene package is selected for system design and test.

5.2.1 Sample space selection

The web pages of Durham University are selected and indexed as the sample space, including the courses information, department introduction, accommodation updates and staff contact details. The system has collected and indexed more than 20,000 web pages from the Durham University website with the Apache Lucene package.

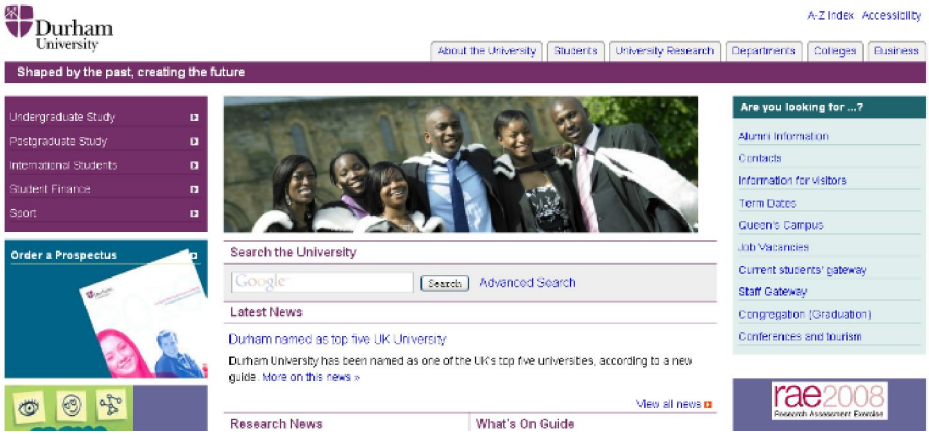


Figure 5.1 Durham University Website

5.2.2 Experiment Platform

For this preliminary set of experiments with 20,000 web pages, information is collected and processed by the Lucence indexer. The indexer creates an index file and stores keyword frequencies information in a table and runs search function against these web pages.

Figure 5.2 shows the experiment platform developed based on the Lucene package. The platform provides powerful and sophisticated functions to search in the following indexed 20,000 web pages.

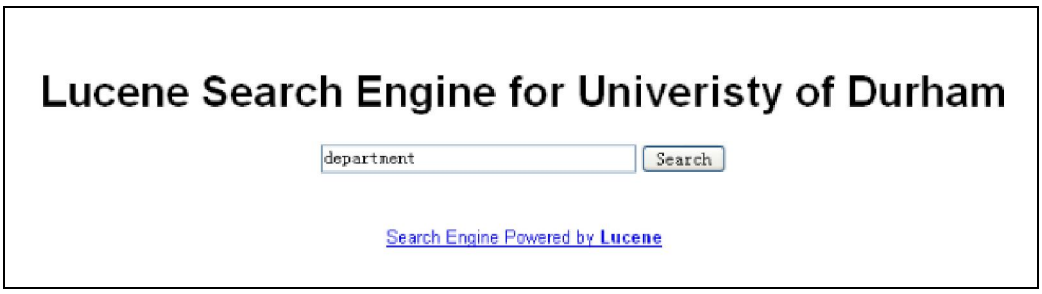


Figure 5.2 Lucence based search engine for University of Durham

5.2.3 SHOE (Simple HTML Ontology Extensions) Ontology

SHOE (Simple HTML Ontology Extensions) is developed by University of Maryland and it designed to give web pages semantic meaning according to different category (SHOE introduction, 2002). Currently SHOES Ontology includes Beer Ontology, Commerce Ontology, Computer Science Ontology, Personal Ontology and University Ontology, etc.

The reason why we use SHOE here is because that the word selection follows the principles suggested by the SHOE taxonomy. We changed the university ontology to fit Durham University because most of SHOE ontologies are developed based on the academic structure in the America. The following taxonomy is the collection of categories declared in the ontology. The hierarchical form is demonstrated to show the relationship of different elements within the university. In the ontology below, categories followed by an asterisk can be defined in other ontology, for instance, Employee, Education Organization and Book etc.

```
Employee*
    Faculty
      Professor
      Lecturer
      PostDoc
    Assistant
      Research
     Teaching
    AdministrativeStaff
   Director
      Dean
     Clerical (Staff)
      SystemsStaff
Student
    UndergraduateStudent
    PostgraduateStudent
Organization*
  EducationOrganization*
    Department
          Institute
          Program
          ResearchGroup

Publications*
  Article*
    Book*
    Paper*
    Journal*
Book*
   Periodical*
    Journal*
    Magazine*
  Proceedings*
  Thesis*
    DoctoralThesis*
    MastersThesis*
Work*
  Course
  Research
  Schedule
  Conference
```

For example, if we want to find an undergraduate student whose name is “Julian” in Durham University, we can locate web pages for all x and y such that x is a student, y is an undergraduate.

Here are the relationships between the arguments, for example, the student and professor, organization and person, document and person. These arguments can be a type or a category. The relationships below are modified based on the University ontology extension from SHOE project.

```
advisor(Student, Professor)
  affiliateOf(Organization, Person)*
  affiliatedOrganization(Organization, Organization)*
  alumnus(Organization, Person)*
  containedIn(Document, Document)*
  doctoralDegreeFrom(Person, University)
  emailAddress(Person, .STRING)*
  head(Organization, Person)*
  listedCourse(Schedule, Course)
  mastersDegreeFrom(Person, University)
  member(SocialGroup, Person)*
  name(base.SHOEEnity, .STRING)*
  offers(University, Course)
  publicationAuthor(Document, Person)*
  publicationDate(Document, .DATE)*
  publicationOrg(Document, Organization)*
  publicationResearch(Publication, Research)
  publisher(Document, Organization)*
  researchInterest(Person, Research)
  researchProject(ResearchGroup, Research)
  subOrganizationOf(Organization:"suborganization",
    Organization:"superorganization")*
  subject(Document, .SHOEEntity)*
  takesCourse(Student, Course)
  teacherOf(Faculty, Course)
  teachingAssistantOf(TeachingAssistant, Course)
  tenured(Professor, .TRUTH)
  undergraduateDegreeFrom(Person, University)
```

Take the same example above, if we want to find a third year undergraduate student whose name is “Julian” in Durham University studying law degree, we can locate web pages for all x and y such that x is a student, y is undergraduate Where: firstName(x, “Julian”) and Major(x, “law degree”) and Year(x, “third year”) and department(y, “law department”)

5.2.4 Experiment Process

The process to extract the keyword lists from the sample space can be done either manually or automatically. The first phase result is a list of hits returned by the search engine, called List 1. For any second time of search, the keyword submitted by the users from the web browser will firstly be used to match the keyword from List 1. The second word is another central word and again a new list of hits will be returned by the search engine, called List 2.
The principles are set for the words selection from List 1. For example, if a new word provided is close to the first word from the result of first time search, it should appear at the position around the central word, either in front of it or behind it.
Any new central word can be generated from the combination of word 1 or from the most recent central words. This newly generated word can be used to search against List 1 or List 2.
Meanwhile, the central words such as word 1 and word 2 can be used for search in the third time. After a number of word selection processes, the lists of words are achieved. According to the theory, no matter how many words to be added into the list, the search engine will return the similar content of hits and similar amount of it. The list of words is the description of all the main attributes for the information to be found. Figure 5.3 below describes the extraction process for the semantic model.

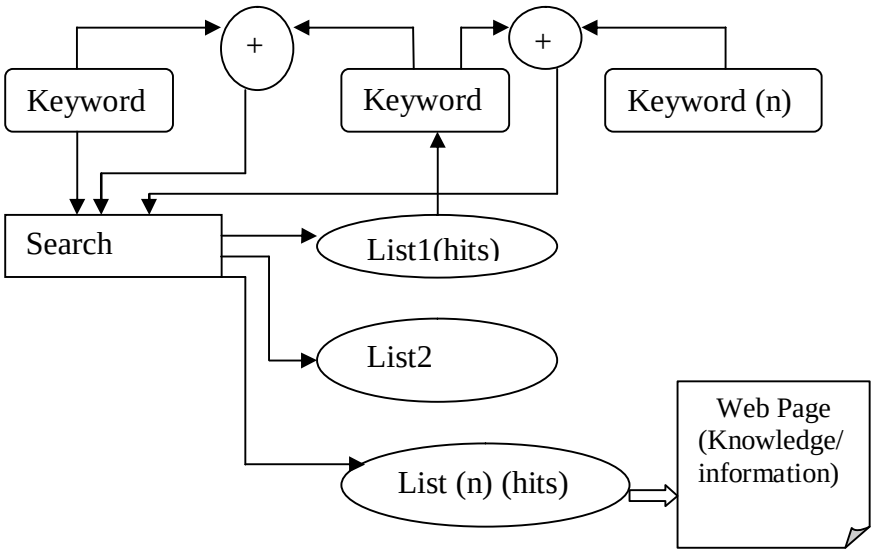


Figure 5.3 Experiment process of Search Engine

For example, a user wants to locate the web page of a lecturer whose name is John and works in the Department of History. Assume the first keyword the user provides is “history”, which can be called as the first central word. This word “history” can be a narrative event, stories or a branch of knowledge that records and analyzes historic events. It can also be a drama based on history events. Thus, about 1,000 hits will be returned, including history story about University of Durham, history courses and

department of history. The result we stored is a pair: K1 – List 1, i.e., history - L1.
From these returned hits, “department” is chosen to be the second word for searching. “Department” is a division of school or college specializing in a particular field of knowledge. “Department of history” is a division of university that researches and teaches history. After this keyword is submitted to the system, the results about “department of history” are returned. We set a second pair: K2 – List 2, i.e. department – L2.
Based on these returned hits, “lecturer” and “John” are chosen to be the third and fourth words, and we get third and fourth pairs: K3 – List3, and K4 – List4.
Hypothetically, let’s assume that there is only one record existed for this lecturer John at department of history. After the above process is going on, when more new words or phrases are added to narrow down the areas, the amount of hits returned will always be the same for the description of John. This group of words {history, department, lecturer, John} is considered as keywords to describe John and as the minimum set of keywords that can be used to identify the unique semantic meaning of the web pages about John.

5.2.5 Keywords Analysis

As we can see in this table, the keywords list “person, employee, lecturer, professor” initially had 153 pages as returned results. After adding more keywords to the search criteria, then the number of the returned pages is shrinked to 22.

Keywords List	RD2	RD3	RD4	RD5	RD6
{Person, Employee, Lecturer, Professor}	153	35	31	22	
{Department, Institute, Research, Group, University}	1370	658	73	14	14
{Student, Undergraduate, Postgraduate}	520	124	95		
{Article, Book, Paper, Journal}	167	115	53	35	
{Book, Periodical, Journal, Magazine}	181	128	65	17	
{Work, Research, Course, Schedule}	408	275	137	59	
{Assistant, Research, Teaching}	385	136	119		

5.3 Experiment Two

5.3.1 Sample space selection and experiment platform

For the problem that some web documents are not well structured, the experiment two is designed here based on a new test sample data that includes groups of PDF documents, while the test samples are all text based in last experiment one. In the experiment with the new extended test space for PDF documents, the theory of the smart search will be analyzed again and some possible search behaviours for the experiment will be discussed.

The platform for the experiment two will be moved from Apache Lucence based search engine to Google platform and its API package. Compared to the experiment one, the process of the experiment two is quite straightforward. Firstly, the words are collected according to the title, author, and description of PDF documents and then input these words in Google. The numbers of the return results will be recorded. To locate any single PDF document, some combination of keywords will be compulsory for the Google platform to search while for each document, the minimum set of keywords can be found to identify each unique PDF document on the web. This is similar to the smart search theory discussed in the last chapter. After the keywords set is collected, it will be analyzed and the semantic model will be extracted. The experiment two concentrates more on the results and statistics analysis.

The Google are selected as the sample space including 8,168,684,336 web pages (Google) and PDF documents are selected randomly from Internet and word sets are selected from the online papers according to their appearance order.

5.3.2 Theory in Experiment two

After the first experiment has been done, there are two searching behaviours found possible and described in this chapter.
For desktop search, the document's location and further details are needed to locate the information that is needed. However, it is not sure whether the information existed on web pages or not.
Let PKeyword(num)=M (M is equal to all the tokenized keywords of title and abstract)
Let Keyword(list)= ={Keyword(1)(title), Keyword(2)(title), Keyword(3)(title), Keyword(4)(title)…Keyword(n)(title), Keyword(n+1)(abstract), Keyword(n+2)(abstract), Keyword(n+3)(abstract)… Keyword(n+n)(abstract)}
Keyword(num)= 2n
Then 1<=Keyword(num)<<PKeyword(num)

Let SP(num)= 8,168,684,336 (Simple space number is 8,168,684,336 web pages)

Let Hit(num)= $\sum_{n=1}^{n} Hit(Keyword(2n))$ = Hit(Keyword(1)(title)＋Keyword(2)(title)＋Keyword(3)(title＋Keyword(4)(title)＋…＋Keyword(n)(title)＋Keyword(n+1)(abstract)＋Keyword(n+2)(abstract)＋ Keyword(n+3)(abstract)＋…＋Keyword(n+n)(abstract))
Then 1<=Hit(num)<<SP(num)

5.3.3 Experiment two's Process

The experiment is taken place following the next five steps described here.

1. Title, author and abstract are extracted from a group of PDF documents
2. Keywords from the title, author and abstract are tokenized. Here, only title and author are used and a list of tokenized keywords for each PDF document is extracted as follows:

   {Keyword(1)(title), Keyword(2)(title), Keyword(3)(title), Keyword(4)(title)…Keyword(n)(title), Keyword(n+1)(abstract), Keyword(n+2)(abstract), Keyword(n+3)(abstract)… Keyword(n+n)(abstract)}

   (*Keywords are selected from the paper according to their appearance order.)
3. Stop words are discarded from the keywords for each PDF document.
1. Adopt the searching pattern described in figure4.3.4a below to search. Firstly, Keyword (1) is selected to search using Google and the numbers of returned hits are recorded as Hits (1) to analyze. Secondly, Keyword (1) and Keyword (2) are selected to search for a second time to get the return Hits (2). Thirdly, Keyword (1), Keyword (2) and Keyword (3) are selected to search for a third time and the

number of returned hits are recorded in Hits (3).

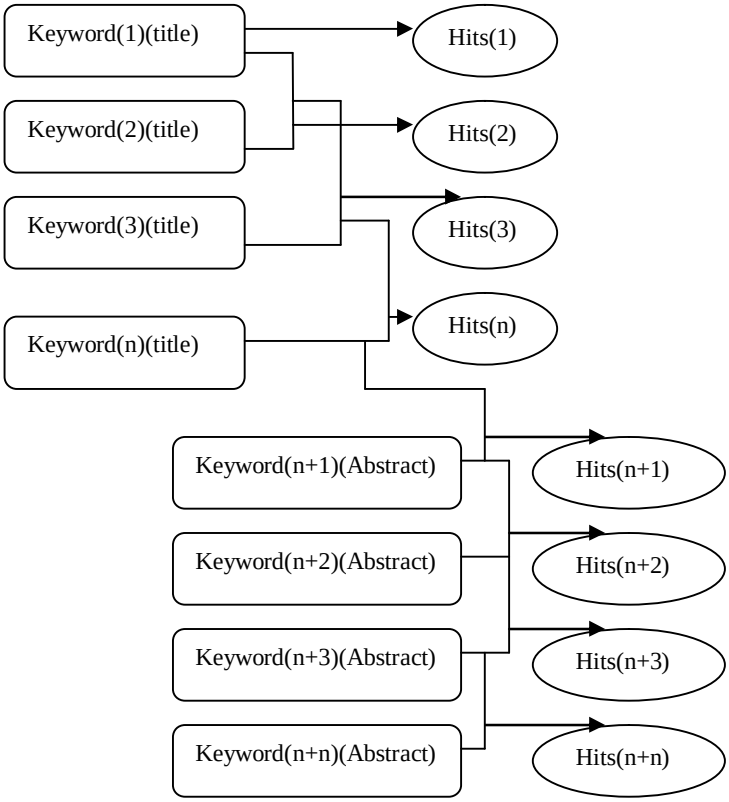


Figure 5.6 Searching pattern

2. Repeat the step 2, step3 and step4 until the search result is stable. Here are the records of returned hits for analysis illustrated below. The analysis process to achieve this is introduced in chapter 5.1. After a number of times of searching like this, the numbers of returned hits will be stay unchanged or only slightly changed based on the smart search theory.

5.4 Comparison of Two Experiments

In this section we compare two experiments to identify the advantages and disadvantages from different point of views including test sample space, experiment process itself and the analysis of these two experiments' results.

1. testing sample space

The testing space of the first experiment is the Website of Durham University. After submitting the keywords and running the search for a few of times, the number of result returned by the Lucene based search engine is stable. The feedback of the experiments is analysed against the smart search theory. Then the larger testing space is chosen from websites of Durham University to information held on Google

platform. Compared to the experiment one that mainly process text based information, the second experiment focus on PDF documents collected from the web.

2. Process of experiment

In the first experiment the keywords for search are chosen randomly to extract a minimum set of keywords to identify unique semantic model from the web document. For the second experiment, the keywords are extracted from online PDF documents and information including title, author, abstract and part of main content is collected. Then words are separated into different groups and new lists of keyword set are created. Eventually a minimum set of keywords will be achieved after the search experiment.

3. Analysis of experiment

The analysis of experiment results from two experiments shows confidence in the smart search theory that the numbers of returned hits will stable and a minimum set of keywords exists to uniquely identify any web document.

5.5 The smart search Engine

The objectives of the smart search engine are to provide a stand along semantic search engine so that users can quickly find the specific search results without needing to become search engine experts. The smart search Engine combines keyword-based search engine with web extractor and ontologies to navigate the search results. The semantic structure used in the smart search engine is generated semi automatically using the algorithm explained in chapter4.

The main components in the smart search engine include the webpage indexer, the information extractor, and the search engine. The webpage indexer and the information extractor perform the operation of a traditional search engine based on keywords.

The smart search engine is very similar to a traditional search engine but the results are more accurate and efficient, the following samples will explain how the smart search engine works.

Figure 5.4 the smart search Engine Portal

The following picture shows the web interface of the smart search engine for University of Durham. Users can enter some keywords; the smart search engine is able to retrieve results containing those keywords. For example, a user enters “department” in the smart search engine, the user not only gets syntactically matching results (with the keyword “department” in the results, for instance, “history department”, “geography department” etc.) but the query is analyzed by the smart search engine according to its meaning. The smart search engine can access the ontology built for Durham University and try to identify the concepts that connected with the query. In the previous example, the smart search engine can identify the keyword “department” and give all related keywords the organizational structure. (“institute”, “program”, “research group”, “school”, “university”)

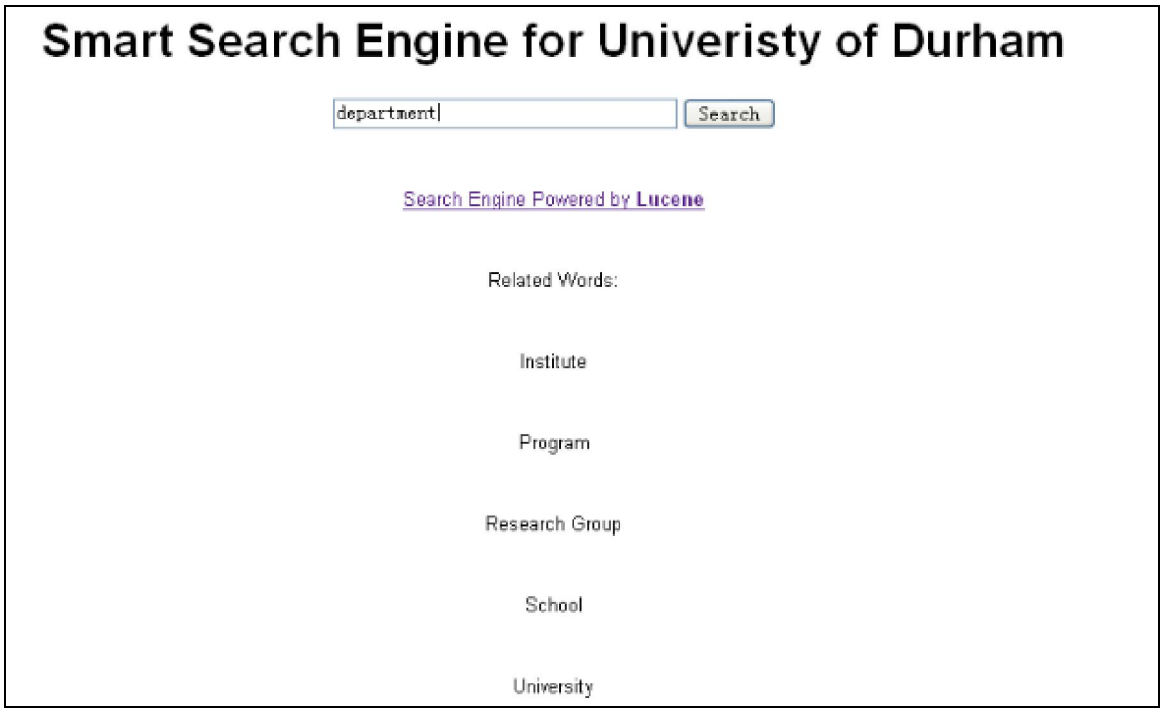


Figure 5.5 Web interface

The user can manually select a word from the related words list. The smart search engine will return more precise results. It can not only provide related words but also relationships between words. These are relations between concepts identified in ontologies.

Chapter6. Analysis and Conclusion

In this chapter the evaluation of these two experiments will be carried out and the result and analysis will also be concluded as achievement for this research project. Based on the smart search theory, the semantic model is extracted to describe the entity.

6.1 Analysis of Experiments

The purpose of experiment analysis is to verify the semantic model extracted to uniquely identify a single web document. In this section a description will be given regarding the preparation of data and its analysis process.
The processes of data analysis are as follows

1. Data Preparation Phase

The testing space is chosen from 8,168,684,336 web pages that Google indexed (Schmidt, 2005) and 200 PDF documents are collected from these web pages randomly. All these documents will be divided into ten different groups.

2. Data Analysis Phase

After test data is prepared, statistics of a minimum set of keywords can be then achieved by testing with different keywords. As illustrated in figure 6.1, the numbers of returned results tends to be stable when a combination of keyword set is generated, which matches the theory proposed in chapter 4.

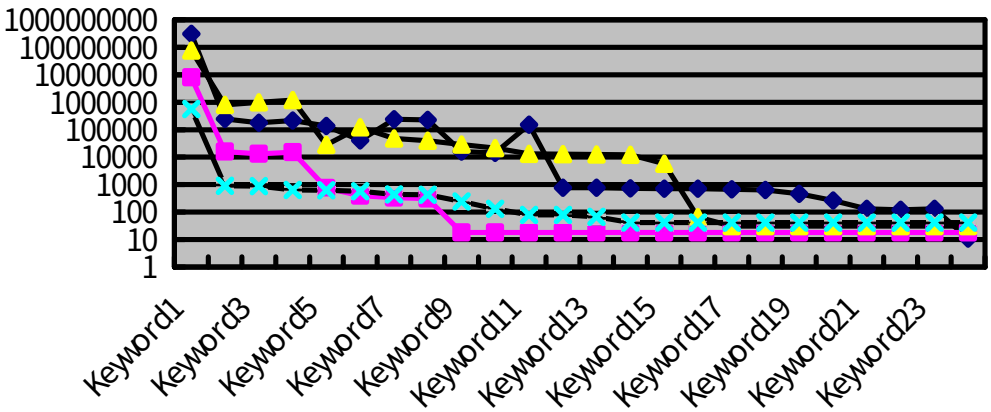


Figure 6.1 Statistics of minimum set of keywords

In this figure above Coordinate X stands for the numbers of keywords provided and submitted by the users to search online PDF documents. Coordinate Y indicates the amount of results returned based on the keywords submitted. It is explicitly illustrated that when more keywords are provided to refine the search criteria, the returned results tends to be more precise and the amount of it will hit certain level when the content is clearly described. The testing space is 10,000,000 of web documents (including websites, PDF documents, etc.) After submitting with two keyword words for first trial search, the number of returned results is nearly down from 10,000,000 to 10,000. The second word is separate keyword in this experiment. After adding word 3 and word 4 into the search criteria, the results returned by search platform are barely changed, that is, the word 3 and word 4 can not be listed as keywords. The word group that is chosen for this experiment meets the following principles:
Group1 contains the information of Title, Group2 for Description; Group3 for Abstract; Group4 for main body. The combination will consist of these four groups.

6.2 Conclusion

In this project the main problems in the research area of information retrieval and semantic search are addressed and it proposes the smart search theory as new theory based on hypothesis that semantic meanings of a document can be extracted to a set of keywords.

The objective of the smart search engine is to provide a semantic search engine so that users can quickly find the specific information and results without being engine experts. The smart search engine combines keyword-based search engine with web extractor and ontologies to navigate the search results. The semantic structure used in the smart search engine is generated semi automatically using the algorithm explained in chapter 4. The main components in the smart search engine include the webpage indexer, the information extractor, and the search engine. The webpage indexer and the information extractor perform the operation of a traditional search engine based on keywords. The smart search engine is similar to a traditional search engine but the results are more accurate and efficient.

With two experiments designed and carried out in this project, the experiment result demonstrates positive facts that meet our smart search theory.
In the theory proposed in this project, the smart search aims to conclude a set of keywords for any web document, by which the semantic meanings of the documents can be uniquely identified. Meanwhile, the size of the set of keywords is supposed to be small enough which can be easily managed. This is the fundamental assumption for creating the smart semantic search engine. In this project, the rationale of the assumption and the theory based on it is discussed, as well as the processes of how the theory can be applied to the keyword collection. Then the design of the smart search engine is proposed to create a solution to ease the efficiency problem while searching large among of information on the web.

To achieve high efficiency in web searching, using statistical methods is proved to be an effective way, which can be interpreted from the semantic level. Based on the frequency of joint keywords, the keyword list can be generated and linked to each other to form a meaning structure.

Based on these two experiments, keyword list is extracted from the web documents and some of these can be combined to the university ontology. The fundamental key of the smart search is to extract and locate the relationships of different elements in a web document.

According to the theory proposed in the thesis that any web document can be uniquely identified by using a set of keywords, my contribution is to locate the keywords for the smart search engine. The first step is narrowing down the keywords lists which identify those web documents. The key idea to use the SHOES ontology to find an attribute list from a list of keywords we prepare in advance for the test space within a category. The two experiments are designed to demonstrate the smart search theory and the hypothesis addressed. A semantic relationship model is used for semi-structured or unstructured information on the World Wide Web to help improve the accuracy and efficiency of smart search engine.

The semantic structures concluded from our two experiments here can be applied to any web documents to be identified by a minimum set of keywords. The purposes of the experiments are to extract the semantic model and evaluate the appropriateness of the semantic relationship model extracted from web page.

6.3 Future work

These two experiments only reflect small parts of a real word. The future work is to improve the intelligent smart search engine developed based on the smart search theory proposed in this project. These two experiments demonstrate and reflect the smart search theory quite well in terms of web documents being identified and located by a minimum set of keywords. To make the experiments and the results more convincing statistically, large amount of information need to be indexed as the test sample space to search, compared to the fact in the first experiment that only 20,000 web pages are indexed as the experimental sample space within Durham University web site.

In experiment two, two attributes (title and keyword) are used to extract keyword list by using the SHOES ontology for attribute list. More attributes should be considered during the process, i.e. abstract, author and content, etc. Besides, the keyword list extracted from web documents are based on their random appearance but for the future work, it would be extracted based on different keyword order. When enough analysis is done, the new search platform could return users with high preciseness of results.

Reference

Allemang, D. & Hendler, J. (2007) *Semantic Web for the Working Ontologist: Effective Modelling in RDFS and OWL*. London: Morgan Kaufmann Publisher.

ALVIS – Superpeer Semantic Search Engine Executive Summary. (2005) Available at: ftp://ftp.cordis.europa.eu/.../alvis-executive-summary-2005_en.pdf
(Accessed: Feb 2008)

Antoniou, G. & Harmelen F, V. (2008) *A Semantic Web Primer*. 2nd edition. Cambridge, Massachusetts, London, England: The MIT Press.

Ardo, A. (2005) *Focused crawling in ALVIS semantic search engine*. Available at: http://www.it.lth.se/knowlib/publ/eswc2005posterardo.pdf
(Accessed: Feb 2008)

Baeza-Yates, B. & Ribeiro-Neto, B. (1999) *Modern Information Retrieval*. Addison: London.

Baloglu, A. & Abdel-Badeeh, M. (2007) 'Intelligent Web- Based System: A Proposed Model for E-Lab Services', *International Journal of Soft Computing Applications*, No.2, pp.5-14.

Berners-Lee, T. Hendler, J. & Lassila, O. (2001) 'The Semantic Web', *Scientific American*, 284(5), pp. 34-43.

Blachman, N. & Peek, J (2003) *Google Guide: How Google Works*, Available at: http://www.googleguide.com/google_works.html (Accessed: Dec 2008)

Buntine, W. Valtonen, K. & Taylor, M. (2005) *The ALVIS Document Model for a Semantic Search Engine*, Available at:
http://eprints.pascal-network.org/archive/00000976/01/buntineESWC05.pdf
(Accessed: Dec 2008)

Brin, S & Page L (1998) *The anatomy of large-scale hyper textual web search engine*, Computer Science Department, Stanford University, Available at: http://infolab.stanford.edu/pub/papers/google.pdf
(Accessed: Nov 2008)

Chaudhuri, S. Das, G. Hristidis, V. & Weikum, G. (2006). 'Probabilistic information retrieval approach for ranking of database query results', *ACM Transactions on Database Systems (TODS), 31*(3), pp. 1134 –1168.

Claburn, T (2008) *Google index reaches 1 trillion URLs*, Information Week, Business Technology Network, Available at: http://www.informationweek.com/blog/main/archives/2008/07/google_index_re.html
(Accessed: Dec 2008)

Clarke, C & Cormack G (1995), *Dynamic Inverted Indexes for a Distributed Full-Text Retrieval System*, TechRep MT-95-01, University of Waterloo, February 1995.

Collier, K. et al. (1998) 'A Perspective on Data Mining', *Centre for Data Insight*, Northern Arizona University, USA, pp 4-5.

Cutting, D (2006) *Apache Lucene: Index File Format.* Available at: http://lucene.apache.org/java/2_1_0/fileformats.pdf
(Accessed: Jan 2009)

Davies, J, Studer, R And Warren, P. (2006) *Semantic Web technologies*. Trends and research in ontology-based systems: Wiley edition, England.

Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R., Peng, Y., Reddivari, P., Doshi, V. & Sachs, J. (2004) *Swoogle: A Search and Metadata Engine for the Semantic Web.* Available at: http://ebiquity.umbc.edu/_file_directory_/papers/115.pdf
(Accessed: May 2008)

Fayyad, U., Piatetsky-shapiro, G. & Smyth, P. (1996) 'Knowledge Discovery and Data Mining: Towards a Unifying Framework', *The Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, AAAI Press, USA.

Goetz, D. (2000) *The Lucene search engine: Powerful, flexible and free.* Available at: http://www.javaworld.com/javaworld/jw-09-2000/jw-0915-lucene.html?page=2
(Accessed: Jan 2009)

Grossman, D & Frieder, O. (2004) *Information retrieval: Algorithms and heuristics.* New York, NY: Springer.

Gruber, T. (1993), "A Translation Approach to Portable Ontology Specifications", *Knowledge Acquisition*, 5(2), 199-220, 1993.

Hofmann, M. (2003) *The Development of a Generic Data Mining Life Cycle.* Available at: http://www.comp.dit.ie/smckeever/MSc/docs/thesis_final.pdf
(Accessed: Sept 2008)

Huffman, S. (2008) *Search evaluation at Google.* Available at: http://googleblog.blogspot.com/2008/09/search-evaluation-at-google.html
(Accessed: Jan 2009)

Jeffrey, S. & Hunter, J. (2006) *A Semantic Search Engine for the Storage Resource Broker*. Available at: http://dart.edu.au/publications/global-grid-forum-2006-semantic-workshop.pdf
(Accessed: Jan 2009)

Juneja, E. & Phull, E. (2007) *Data Mining and its Scope.* Available at: http://www.rimtengg.com/coit2007/proceedings/pdfs/109.pdf
(Accessed: Feb 2009)

Lancaster, F, W. (1968) *Information Retrieval Systems: Characteristics, Testing and Evaluation.* Wiley: New York.

Midwinter, P (2007) *Is Google a Semantic Search Engine?* Available at:
http://www.readwriteweb.com/archives/is_google_a_semantic_search_engine.php
(Accessed: Jan 2009)

Moura, E (2005) *Improving Web Search Efficiency via a Locality Based Static Pruning Method*. Available at: http://www2005.org/cdrom/docs/p235.pdf
(Accessed: Feb 2009)

Nakov, O & Bekiarov, L. (2008) *Improving the Search Efficiency of Unstructured P2P Networks by Differentiated Search Algorithm.* Available at: http://csconf.org/Volume2/page426.pdf
(Accessed: Feb 2009)

Ounis, L. Amati, G. Plachouras, V. He, B. Macdonald, C & Lioma, C (2006) *Terrier: A high performance and scalable information retrieval platform.* Available at:
http://ir.dcs.gla.ac.uk/terrier/publications/ounis06terrier-osir.pdf
(Accessed: Jan 2009)

Paul, T (2004) *The Lucene Search Engine: Adding search to your applications.* Available at: http://www.javaranch.com/journal/2004/04/Lucene.html
(Accessed: Jan 2009)

Petratos, P. (2006) 'Information retrieval systems: A perspective on human computer interaction', *Journal of Issues in Informing Science and Information Technology*, *3*, pp.511-518. Available at
http://informingscience.org/proceedings/InSITE2006/IISITPetr231.pdf

RDF/XML Syntax Specification. (2004) W3C Recommendation,
Available at: http://www.w3.org/TR/rdf-syntax-grammar/
(Accessed: Jan 2009)

Rijsbergen, V. (1979) *Information Retrieval*. 2nd edition. Butterworth-Heinemann.

Salton, G & McGill, M, H. (1983) *Introduction to Modern Information Retrieval*. McGraw-Hill.

Schmidt, E. (2005) *How many pages in Google? Take a guess.* Available at: http://www.nytimes.com/2005/09/27/technology/27search.html
(Accessed: Feb 2009)

Search Engine Index (2008) Available at:
http://en.wikipedia.org/wiki/Index_(search_engine)#cite_note-0
(Accessed: Nov 2008)

Semantic Web (2008) Available at: http://en.wikipedia.org/wiki/Semantic_web (Accessed: Nov 2008)

SHOE introduction (2002) Available at: http://www.cs.umd.edu/projects/plus/SHOE/ (Accessed: May 2009)

Staikos, K. (2000) *The great libraries: From antiquity to the Renaissance, 3000 B.C. to A.D.1600*. New Castle, Delaware: Oak Knoll Press.

Stopford, B. (2006) *The Topic Specific Search Engine.* Available at: http://www.benstopford.com/articles/The%20Topic%20Specific%20Search%20Engine.pdf
(Accessed: Jan 2009)

Tharanga, W. M & Ranasinghe, D (2006) *Search Engine: An Effective tool for exploring the Internet*, Library of Eastern University of Sri Lanka, Available at: http://dlist.sir.arizona.edu/1623/01/Internet_Search_Engines.pdf
(Accessed: July 2008)

Web Crawler (2008) Available at: http://en.wikipedia.org/wiki/Web_crawler (Accessed: Nov 2008)

Web Search Query (2008) Available at: http://en.wikipedia.org/wiki/Web_search_query
(Accessed: Nov 2008)