Enhancement of Weight Calculation in Ranking of Internet Search Engines

By

Hisham Khaleel Hamed Abu Jalban

Supervisor

Dr. Moayad A. Fadhil

This Thesis was Submitted in Partial Fulfillment of the Requirements for the Master's Degree in Computer Science

Deanship of Academic Research and Graduate Studies

Philadelphia University

December-2008

جامعة فيلادلفيا

نموذج تفويض

أهشا ام خلي ل حام د أب و جلب ان ،أف وضجامع ةفيلادلفي ابتزوي دنس خم نرس التيللمكتب ات أو المؤسسات أو الهيئات أو الأشخاص عند طلبها.

التوقيع :

التاريخ :

Philadelphia University

Authorization Form

I, Hisham Khaleel Hamed Abu Jalban, authorize Philadelphia University to supply copies of my thesis to libraries or establishments or individuals upon request.

Signature:

Date:

Committee Decision

Successfully defended and approved on ____________________________

Examination Committee   Signature

Dr____________________________________ Chairman

Academic Rank: ________________________   ______________

Dr. ___________________________________ member.

Academic Rank__________________________   ______________

Dr. ___________________________________ member.

Academic Rank:_________________________   ______________

Dr. __________________________ External Member.

Academic Rank:_________________________   ______________

Acknowledgment

I would like to express my deepest regards to my supervisor and mentor whose suggestions and guidance led to presentation of this thesis.

I also wish to express great admiration and love to my parents and wife for their encouragement, support and patience through my studies.

I wish to send my ultimate thanks given to my professors who taught me in the last two years.

Hisham Jalabneh

Table of Contents

List of Figures

List of Tables

List of Abbreviations

No.	Abbreviation	Stands For
1	ARPA	Advanced Research Projects Agency
2	HITS	Hyperlink Induced Topic Search
3	HTML	Hyper Text Markup Language
4	IDF	Inverse Document Frequency
5	IR	Information Retrieval
6	ISAM	Index Sequential Access Mode
7	ISP	Internet Service Provider
8	NASA	National Aeronautics and Space Administration
9	NSF	National Science Foundation
10	SE	Search Engine
11	TCP	Transport Control Protocol
12	TF	Term Frequency
13	URL	Unified Resource Locator
14	WSE	Web Search Engines
15	WWW	World Wide Web
16	WWWW	World Wide Web Worm

List of Algorithms

ABSTRACT

This thesis proposes new factors for indexing documents; enhance the weight calculations for the ranking algorithm and combining the advanced link-based ranker with the term-based ranker, in order to enhance the ranking score for documents, which led to improve the recall and precision of the search engine and for making more relevant documents appear at the beginning of the results list.

The proposed system which is made especially to test the new enhancements on indexer and ranker has been tested and evaluated, with various queries, and used to compare the advanced link-based page ranker with the original link-based ranker and term-based ranker.

The relevance of the returned documents (recall, precision) has been considerably improved in comparison between the advanced link-based ranker with original link-based ranker and term-based ranker by more than 13% on queries' results tested on more than 3000 Web pages.

Chapter One

Introduction

1.1 Internet

The Internet is probably the largest revolution in the computer industry since the advent of the personal computer (PC) and is a valuable clinical tool for physicians and other health care personnel. It can be used to communicate with colleagues around the world, to obtain information (including practice guidelines, abstracts, and journal articles), and to arrange travel and meetings.

Although the technology on which the Internet is based was developed in the mid-1960s, it was not widely available until almost 1990, and access was primarily limited to universities, government agencies, or the computer industry. Until recently complex software and restricted access have required Internet users to have a sophisticated understanding of computers, networking, and programming. New technology has, however, made the Internet available to nearly anyone with access to a PC and a modem (a device that connects the PC to other computers over telephone lines or wireless).

Getting information on the Internet is now as easy as inserting a disk, clicking a "Setup" icon, and then pointing to a topic with a mouse. Sophisticated multimedia documents, including video, sounds, and pictures, and which cover every topic from fibre-optic intubations to the stock market, are accessible nearly anywhere in the world.

The Internet's history begins in the mid-1960s, during the Cold War, at which time the United States Department of Defence relied on a network of powerful super-computers to control ballistic missiles and other weapons. It was determined that in the event of a war, it might be necessary to change this network rapidly or add new computers on short notice (e.g., by parachuting computers into an area and connecting them by radio). As a result, the military, through the Defence Advanced Research Projects Agency (ARPA), began to investigate new simple and reliable ways to connect computers Doyle DJ et al (1996).

1.2 Internet Protocols

The central assumption of this new technology was that physical connections were unreliable, i.e., a connection could be lost at any time. To avoid this problem, each computer would keep a constantly updated list of its neighbours and would be able to find alternative pathways to a particular destination if a connection were severed.

To provide reliable communication in this environment, two closely interacting protocols were developed: the "Internet Protocol" (IP), which moves small packets of information from one computer to another and the "Transport Control Protocol" (TCP), which breaks large blocks of data into small chunks and reassembles them on the other end. These intertwined protocols are commonly referred to as "TCP/IP."

Personnel at institutions connected to the ARPA Net (the Department of Defence TCP/IP network) quickly learned that linking of computers could permit research reports, programs, data files, and other information to be shared nearly instantaneously with colleagues at remote locations.

Subsequently other government agencies used the TCP/IP protocol to connect their computers. The National Science Foundation (NSF) created NSF Net, a TCP/IP network to link its five supercomputers and to provide remote users with access to its resources. The National Aeronautics and Space Administration (NASA) created the NASA Science Network Doyle DJ et al (1996).

1.3 Internet Development

By the early 1980s, the biggest obstacle to growth of the Internet was bureaucracy. The US government set strict "appropriate use" policies that governed what information could be sent and how each network could be used. Administrators were encumbered by the regulations and by the careful record keeping that were required. To solve this problem, Congress passed a law combining ARPA Net, NASA Science Net, and NSF Net into the National Research and Education Network, administered by the NSF. The Internet was born.

Researchers and scientists at universities and federal agencies, who soon discovered that TCP/IP networks were easy to connect and that they could expand without disrupting existing networks, were the initial users of the Internet. To promote widespread use of the new network, the NSF created policies that encouraged institutions to make access available to individual users. In 1993, Commercial Internet Service Providers (ISPs) were allowed to sell access to the general public, and the number of Internet users increased rapidly. Shortly thereafter, the development of advanced Internet services, in particular the World Wide Web (WWW), greatly simplified use of the Internet and further increased the rate of growth.

Nearly the entire Internet is currently privately funded and maintained by telecommunications and computer companies, educational institutions, and other organizations; the government now funds only those sections that it uses Doyle DJ et al (1996), Hsinchun Chen et al (1996).

1.4 Search Engines

Due to the massive growth of the Internet, users overloaded with data but in lack for knowledge and information. Systems invited to lead the way and guide the users to proper information, called Search Engines.

These systems, which combine (among others) such disciplines as database technology, distributed computing and storage, statistical linguistics and graph algorithms, are presented with queries of users.

There are many powerful search services on the Web. Such Web search services are Yahoo, Alta Vista, Google, Lycos, Infoseek, Excite, WebCrawler and others. In the Internet world, these search services are called Search Engines.

Search Engines are tools used to help Web users in finding their favorite information or Web sites. It enables various kinds of words or graphics searches. Each of the Internet Search Engines has its own rules and works differently.

Some kinds of Search Engines are composed of huge databases of the Web page files that have been assembled automatically by machine. These databases contain information about Web pages that have registered with a particular Search Engine, such as Yahoo! At Yahoo!, registrations are entered by humans, who categorize entries by subject Hsinchun Chen et al (1996).

Users expect their queries to be instantaneously answered with ranked lists of the most relevant Unified Resource Locators (URLs) available online for each query. In order to meet these demands, Web Search Engines (WSEs) must collect and index billions of resources, and develop highly efficient retrieval and ranking algorithms that are capable of effectively answering queries in almost a blink of the eye Baeza Yates et al (1999), M.W. Berry et al (1999).

Largest WSEs index thousands of millions of multilingual Web pages containing millions of distinct terms. Due to the peculiarities and the huge size of the Web repository, traditional Information Retrieval (IR) systems developed for searching smaller collections of structured or unstructured documents appear inappropriate for granting retrieval effectiveness on the Web.

Most components of an IR system must be rethought in order to address the problem of effectively and efficiently searching and retrieving information from the Web. Consider, as an example, the problem of deciding which documents are relevant and which are not with respect to a user query.

This hard task is commonly delegated to a ranking algorithm which attempts to establish an ordering among the documents retrieved. Documents appearing at the top of this ordering are the ones considered to be more relevant. The two most accepted metrics to measure ranking effectiveness are: ***Precision*** (i.e. number of relevant documents retrieved over the total number of retrieved documents) and ***Recall*** (i.e. number of relevant documents retrieved over the total number of relevant documents in the collection) C.J. Van Rijsbergen (1979). In traditional IR, it can be assumed that the documents in the collection originate from a reputable source and all words found in a document were intended for the researcher.

Ranking can simply be based on statistics performed over word frequencies. The same assumption does not hold on the Web where content is authored by sources of varying quality and words are often added indiscriminately to boost the page’s ranking. Moreover, as the size of the indexed collection grows, since users usually only look at the first few tens of results, a very high precision has to be preferred even at the expense of the recall parameter.

Similar considerations can be done for others key components of an IR system. The size of the Web data repository along with its exponential growth, the heterogeneity and dynamicity of Web data, are all challenging problems justifying the structural complexity of the software architecture of modern WSEs that exploit a variety of novel technologies developed in several related research areas such as databases, parallel computing, artificial intelligence, statistics, etc.

In this research, three different WSE issues will be illustrated. These issues are: ranking of Search Engine query results, indexing, and optimizing Search Engine query.

1.5 Aims of the thesis:

This research has a number of aims:

- Review literature relating to Internet Search Engine technology and in particular information retrieval systems.
- Provide and explain step by step of areas that will lead to full understanding of the Search Engine taxonomy.
- Enhance the indexer of the Search Engine through preprocessing, through classifying new criteria, giving weight to each case and shape of the words and the URLs in the documents; the system will use a combination of Boolean, Vector, and link-based models.
- Enhance the ranking of the retrieved information, through enhancing the weight calculation for the relation between the documents and the query.
- Design, develop, and evaluate a Search Engine for a locally hosted Web site in order to test the enhancement on indexing and ranking algorithms.

1.6 Research Outline

The research contains number of chapters as follows:

Chapter 2: literature Review

This chapter provides an open picture on the articles, topics, and important subjects taken into consideration in the study of Search Engines. It provides a couple of articles which are used as a reference in writing this research.

Chapter 3: Retrieval Models

The topic of information retrieval is addressed in this chapter. Taxonomy of models is identified and a brief explanation of each model in the taxonomy is provided.

Chapter 4: Search Engines

This chapter provides a high level overview of Search Engine technology. It identifies the standard architecture for a Search Engine and provides a detailed discussion on the area of search interfaces. This chapter will also create a link between Search Engines and traditional information retrieval systems, also will give a good idea about Google Search Engine as a prototype.

Chapter 5: Design and Implementation

This chapter presents a high level overview of the Search Engine architecture. Key system requirements are identified and a logical overview of the systems main components is provided.

Chapter 6: Testing and Evaluation

In this chapter the Search Engine is put through testing and evaluation. A selection of test queries is used to test the usability and functionality of the new enhanced system. Results from these tests are then evaluated and the appropriate action is considered.

Chapter 7: Conclusions and Future Works

This chapter reviews the aims and objectives of the research and looks at areas where future work may be appropriate.

Chapter Two

Literature Review

2.1 Introduction

Due to an ever-growing World Wide Web, the volume of data available online increases exponentially day by day, the data resides in different forms, ranging from unstructured data in file systems to highly structured, in relational database systems. Some of this data is raw data, e.g., images or sound. Some of it has structure even if the structure is often implicit, and not as rigid or regular as that found in standard database systems. There are no authorship standards, no editorial board and no imposed topical hierarchy to publish data on the Web.

In this chaotic sea of data, millions of people by the minute from all over the world are starving for information. Web search made it possible, by complex information retrieval systems called ***Search Engine***, which combine such disciplines as database technology, distributed computing, storage, statistical linguistics and graph algorithms, that been presented as queries to users. Users expect their queries to be instantaneously answered related to the most relevant Uniform Resource Locators (URLs) available online for each query.

2.2 Related Work

Sergey Brin, Lawrence Page (1998)

The paper presents Google as a prototype of a large-scale Search Engine which makes heavy use of the structure present in hypertext. Google is designed to crawl and index the Web efficiently and produce much more satisfying search results than existing systems. The research also presents Google system’s features that gives high precision and recall results.

This paper describes the important goals that are achieved by implementing the designed PageRank algorithm, through describing and analyzing Google Search Engine. These goals are:

The main goal is to improve the quality of Web Search Engines. Another important design goal is to build systems that reasonable numbers of people can actually use.

The PageRank algorithm makes use of both link structure and anchor text, and is used as a parameter for measuring page importance by Google Search Engine.

Craig Silverstein et al, (1999)

The paper presented an analysis of an **AltaVista** Search Engine (2008)query log consisting of approximately one billion entries for search requests over a period of six weeks, also presents an analysis of individual queries, query duplication, and query sessions.

The paper presented results of a correlation analysis of the log entries, studying the interaction of terms within queries. The data presented in the paper supports the inference that Web users differ significantly from the user assumed in the standard information retrieval literature. Specifically, it shows that Web users type in short queries, mostly look at the first 10 results only, and seldom modify the query. This suggests that traditional information retrieval techniques may not work well for answering Web search requests.

Krishna Bharat (2000)

The paper describes an extension to Search Engines to explicitly maintain user search context as they look for information, on many topics, using many Search Engines, and over many sessions. The paper presents an extension to Search Engines called ***SearchPad*** that makes it possible to keep track of ‘search context’ explicitly. ***SearchPad*** is an agent that works collaboratively with result pages, and allows users to remember queries and associated leads in a convenient helper window. It describes an efficient implementation of this idea deployed on four Search Engines: AltaVista, Excite, Google and Hotbot.

Arvind Arasu et al, (2002)

The paper presents a simple modification to the PageRank algorithm (ranking algorithm that presented by Page and Brin). By this modification, PageRank algorithm uses the most recent values for every Web page pointing to a Web page.

Chirita P.A., et al (2003)

The article presented a new algorithm for finding hubs and authorities related to a specific Web page. It is a slight modification of the Google PageRank Algorithm. Various aspects have been investigated in this direction, like extensions of the Hyperlink-Induced-Topic-Search (HITS) algorithm or modifications to the PageRank algorithm. The paper focuses mainly on oriented computing hub scores, but the formulas for authority scores are almost always analogous.

Liwen Vaughan, (2004)

The paper proposed two measurements, as counterparts of traditional recall and precision: the quality of result ranking and the ability to retrieve top ranked pages. The main difference between these measurements and those used in earlier studies is that these new measures are based on a continuous ranking of test documents (ranked from the most relevant to the least relevant) rather than the discrete relevance judgments (e.g. relevant, partial relevant, irrelevant) used in previous studies.

The author conducted an experiment to test these new measurements by applying them to a performance comparison of three commercial Search Engines: Google, AltaVista, and Teoma. Results show that the proposed measurements are able to distinguish Search Engine performance very well.

Kyung Joong Kim, Sung Bae Cho (2007)

The paper presented how personalized Search Engines are important tools for finding Web documents for specific users, because they are able to provide the location of information on the World Wide Web (WWW) as accurately as possible, using efficient methods of data mining and knowledge discovery.

The paper showed how to find relevant Web documents for a given user; the proposed Search Engine uses link structures and a fuzzy concept network. The Search Engine finds relevant documents in which user is interested in, and reorders it with respect to the user’s interests. The Search Engine finds authoritative and hub sources for a user query using link structures. For efficient searching, these link structures are stored in advance. The fuzzy document retrieval system personalizes the link-based search results with respect to the user’s interests. The user’s knowledge is represented using the fuzzy concept network.

The paper presented how Future work will proceed, using the user’s feedback about the search results.

Amir Hosein Keyhanipour et al (2007)

The paper presented, that Meta-Search Engines could be considered as an interface on the top of local Search Engines to provide uniform access to many local Search Engines. In the paper, a novel meta-Search Engine, named as Web Fusion, was introduced. Web Fusion learns the expertness of the underlying Search Engines in a certain category based on the users’ references. It also uses the ‘‘click-through data concept’’ to give a content-oriented ranking score to each result page.

The optimum integration of decision lists of Search Engines and awareness of the users’ preferences are the most challenging problems in the area of Meta-Search Engines. The experimental results show that concerning the user preferences as well as integration of decision lists has significant effect on reducing users’ time and effort for finding the required information.

Chapter Three

Retrieval Models

3.1 Introduction

The previous chapter provided the reader with an overview on the background and literature needed to understand topics of Search Engines. This chapter will provide an overview of retrieval models used in modern information retrieval systems and will introduce the reader to a well accepted taxonomy of retrieval models. In addition, this chapter will provide detailed discussion on a selection of these models.

3.2 Retrieval Models Overview

The majority of retrieval systems adopt the use of terms in the document to index and retrieve documents (Note: the term “document” will be used in a general sense to refer not only to text documents but also to any non-textual information, such as multimedia objects). In its simplest form an index term is a set of keywords, mainly nouns, that attempt to describe the semantic content of the document to which it is associated. Although it is widely used, this technique is fundamentally flawed for two reasons.

- Firstly, a lot of the semantic meaning within a document can be lost by representing the full document text with a small subset of terms.
- Secondly, not all the terms that are extracted have the same relevance with regards to the document’s semantic meaning. As a result of this, irrelevant documents are often returned in response to user queries. This leads to frustration on users parts.

Clearly one central problem that affects all information retrieval (IR) systems is the ability to properly predict which documents are to be considered relevant to a particular query and which are not. The job of identifying relevance is usually performed by what are known as ranking algorithms.

Ranking algorithms attempt to order a set of retrieved documents in such a way that those with a high ranking value are considered to be more relevant than those with a lower ranking value. Each of these ranking algorithms will have its own distinct notion as to what constitutes relevance. As a result, these differing notions invariably lead to distinct information retrieval models.

In their book ***"Modern Information Retrieval"*** Baeza Yates et al (1999) attempt to produce a taxonomy of IR models that tries to identify all relevant IR models that have appeared over the years.

3.3 Taxonomy of Information Retrieval Models

Baeza Yates et al (1999) proposed that there are a total of 15 relevant IR models. These can be subdivided into retrieval models and browsing models. Figure 3.1 provides an overview of these models.

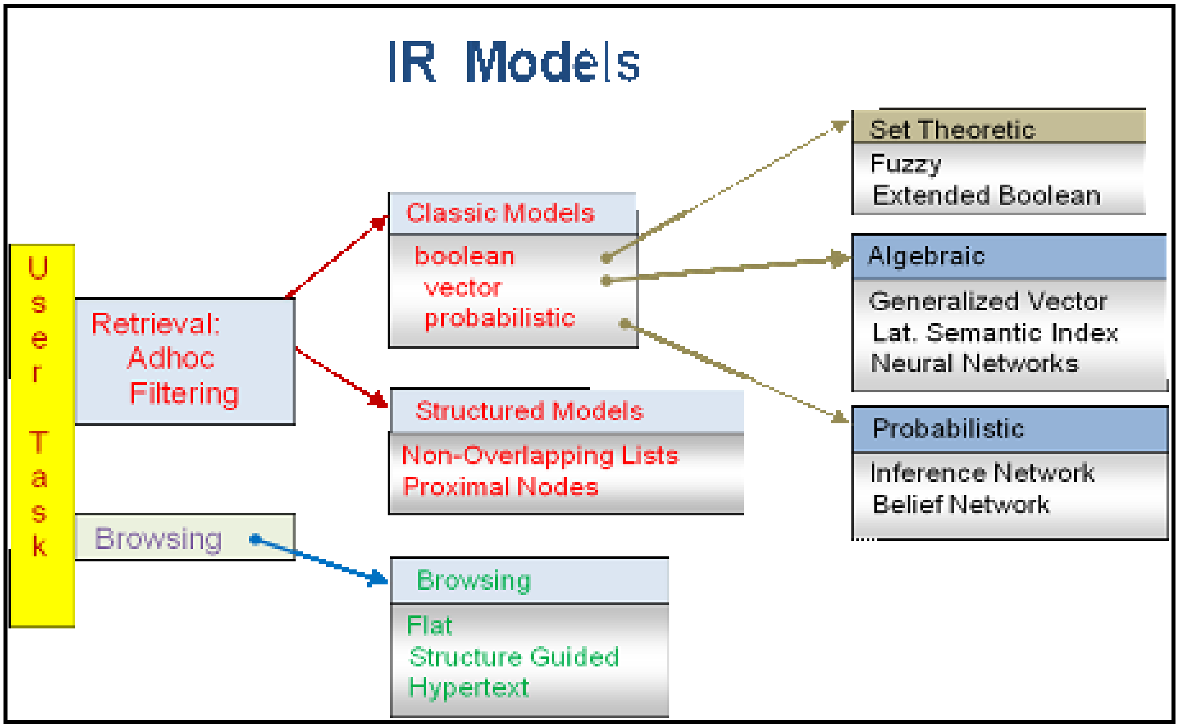


Figure 3.1 : Taxonomy of information retrieval models

The retrieval models are generally associated with systems, in which a user will pose a specific query to the system, and expect a set of relevant documents in return. This model can be subdivided into both the classic and structured models.

Browsing models, on the other hand, do not require that a user pose a specific question. Instead, users generally invest time and effort into browsing the document space looking for relevant references which satisfy or aid them in their search.

3.4 Classic Models

Classic models consider that each document within the system can be described by a set of index terms. Identifying these index terms is not a trivial task; care must be taken to ensure that the selected terms are appropriate for the document they are describing. For instance, a noun which appears frequently across a set of documents is of no practical use as it would result in an unacceptably high number of documents being matched. However, a noun that appears in only half a dozen documents would be more desirable as it would result in more precise matches. Once index terms have been identified some mechanism is needed that will match document index terms with query index terms in order to establish document relevance. The three classic models for achieving this are the ***Boolean***, ***vector*** and ***probabilistic models***.

3.4.1Boolean Models

The Boolean model is based on set theory and Boolean algebra. Although it has become the standard model for both large scale information retrieval systems and on-line systems it suffers from a number of drawbacks. Firstly, users of Boolean systems often find it difficult to construct queries which are effective Baeza Yates et al (1999), Witten.I.H et al (1999).

Many Boolean terms such as **AND**, **OR** and **NOT** have different meaning when used in Boolean logic than they do in natural language. For example, in natural language an ordinary user would expect that a query for documents containing the terms "X **AND** Y" would return documents that contain either or both terms. Therefore the result set should contain more entries than would a search for "X" alone. In fact, if this is what the user wanted the **OR** logical operator should have been used.

This has been found to be a common user mistake. Other problems that users experience include the proper use of parenthesis, especially nested parenthesis, as well as effectively identifying and applying techniques which help in the broadening and narrowing of queries. However, possibly the biggest problem associated with the Boolean method is that a document is identified as being relevant or non-relevant, i.e. there is no concept of a document being a partial match. As a result of this limitation the

Boolean model has been extended to include ideas such as the ***fuzzy*** and ***extended Boolean*** models.

The ***fuzzy*** model considers that each query term defines a fuzzy set to which each document will have a degree of membership. A membership function is then used to determine how closely a particular query matches a document. Usually a value between 0 and 1 will be returned where 0 indicates that there is no match and 1 indicates a full match. This leads to a grading technique which is less abrupt than the standard Boolean technique. Baeza Yates et al (1999), Piccenelli.G et al (2008)

The ***extended Boolean*** model adapts Boolean logic so that operators are treated more or less strictly. For example, a query of the form "X **AND** Y" would indicate that documents which only contain one of the terms should not necessarily be ignored while a query of the form "X **OR** Y" would indicate that documents containing both "X" and "Y" should be considered more appropriate than documents containing only one of the terms.

3.4.2 Vector Models

Even with the extended models described above the Boolean method tends to be imprecise as partial matching of documents is difficult Baeza Yates et al (1999), Jasminka Dosa et al (2008), Wilkinson.R et al (1991). Therefore the vector space model has been proposed which incorporates the idea of partial matches into its framework. This is achieved by assigning non-binary weights to index terms in both queries and documents Baeza Yates et al (1999).

These weights can then be used to compute how similar each document in the system is to the user queries. The concepts behind vector space modelling are that by placing terms, documents, and queries in a term-document space it is possible to compute the similarities between queries and the terms or documents, and allow the results of the computation to be ranked according to the similarity measure between them Braun Loes (2002). Essentially the model works on the principle that both document and query can be understood from the terms they contain. By representing these terms as vectors the semantic relationship between queries and documents in the vector space can be calculated. The similarity between a document and a query is judged by the cosine of the angel between the document and query vector. Extensions to the vector space model include algebraic generalized vector, latent semantic indexing and neural networks Baeza Yates et al (1999), Braun Loes (2002).

3.4.3 Probabilistic Models

The basic idea behind the probabilistic model is that for any given user query there is a set of documents which contain only the relevant documents and no other. This is known as the ideal answer set Baeza Yates et al (1999). The properties of this ideal answer set is characterised by a set of index terms.

Although these index terms are not going to be known to the user at the beginning of the query a good guess at them can usually be made. This guess will result in a preliminary probabilistic description of the ideal answer set.

Further interactions are carried out with the user in order to refine and improve the probabilistic description of the answer set. During this interaction the users basically looks at the retrieved documents and decides which ones are relevant to their query. The system uses this information to refine the search process. Because this is repeated a number of times the query will continually evolve and eventually become the real description of the ideal answer set.

Two alternate approaches to this model are the inference network model and the belief network model. Both of these models are derived from Bayesian (belief) networks which are based on probability theory Baeza Yates et al (1999), Fuhr.N (1992).

3.5 Structured Models

The classic model described above is primarily concerned with the retrieval of information based on the semantic content of its documents. However, there is another set of modelling technique that combines both the semantic content of the document with its structural make-up. These models are called structured models and their power can be seen in the following example.

Suppose a user wished to submit a query to a system in which they knew a document existed which contained the text ‘Information Retrieval’ and also had a Figure labelled ‘Neural Networks’. If the users were to submit a traditional query such as (‘information retrieval’ **AND** ‘neural networks’) it is clear to see that the result set would contain many more documents than that desired by the user. If on the other hand the user could specify structural features such as the fact that ‘neural networks’ should only be looked for as a label of a Figure within the document then we could expect a more refined result set. For example, (‘information retrieval’ **AND** Figure(label(‘neural networks’))) should only return documents that contain the term ‘information retrieval’ and ‘neural networks’ where the term ‘neural networks’ appears as the label of a Figure. The most common forms of structured models are non-overlapping lists and proximal nodes. Baeza Yates et al (1999), Jasminka Dosa et al (2008), Braun Loes (2002).

3.6 Browsing Models

The previous models concentrated on scenarios in which a user poses a specific query to the IR system in order to obtain a set of results. An alternative to this approach, which has become more widespread with the advent of the World Wide Web, is a technique known as the ***browsing model***.

It is sometimes the case that a user has a goal which is less defined than that of users in the previous models; such users are often willing to invest time in exploring a particular document space in order to find topics or references that are of interest to them.

Quite often this involves users reviewing flat or hierarchical lists of documents or may even require random navigation through a set of related hyperlinks. The basic idea with this model is that the user manually chooses the paths of navigation in order to make informed decisions. ***Yahoo*** is one example of a Search Engine that allows such browsing. This model can be subdivided into three separate types; these are flat, structure guided and hypertext models. Baeza Yates et al (1999), Karlgen.J (2008).

Chapter Four

Search Engines

4.1 Introduction

The design and implementation, as well the analysis, of efficient, and effective Web Search Engines (WSEs), are becoming more and more important as the size of the Web has continually kept growing. Furthermore, the development of systems for Web Information Retrieval represents a very challenging task whose complexity imposes the knowledge of several concepts coming from many different areas: databases, parallel computing, artificial intelligence, statistics, etc.

This chapter attempts to outline the overall architecture of Search Engines, provide references to existing Search Engines where relevant, introduce the concept of Search Engines as Information Retrieval (IR) systems, and identify the main problems areas associated with such systems.

The Search Engine has been defined in different ways such as:

“A program that indexes documents, then attempts to match documents relevant to a user's search requests.” MarketingTerms.com (2008)

“Web sites that catalogue other Web sites by topic. By entering your subject or title, you access their database which hopefully provides you with a list of Web sites that give you what you want.” BuzzBoltMedia.com (2008)

The researcher defines the Search Engine as a system that indexes Web documents contents in order to fulfil user’s query with a list of relevant documents.

4.2 Search Engines Anatomy

In its simplest terms a Search Engine is a system that indexes, organises and rates documents with the aim of satisfying user queries. The overall architecture of a Search Engine can be sub-dived into three main categories. These are the ***Crawler Manager***, the ***Index Manager*** and the ***Search Interface*** Marendy.P (2001), Lee Underwood (2008) as shown in Figure 4.1.

4.2.1 Crawler Manager

Web crawlers form an integral part of Search Engines. The crawler manager is an automated piece of software that accesses and downloads Web pages and their associated documents on a regular basis. The aim of the crawler is to build up a map of the internet in order to index and store contents of its pages for future searches. Web crawlers work by following a set of pre-defined Web page links Hsiao.R.L (2008).

While processing these Web pages the crawler takes note of any new hyperlinks on the pages. These hyperlinked pages are then parsed for new links, and so on, recursively. The process of downloading and indexing Web pages is a very complicated and time consuming task. For example, if a Web crawler where to only download one page at a time, covering the internet would take a number of years. Instead, Web crawlers generally download thousands of pages in a single go and process them in parallel The Web Robots Page (2008). The crawler will be discussed in more detail in chapter five.

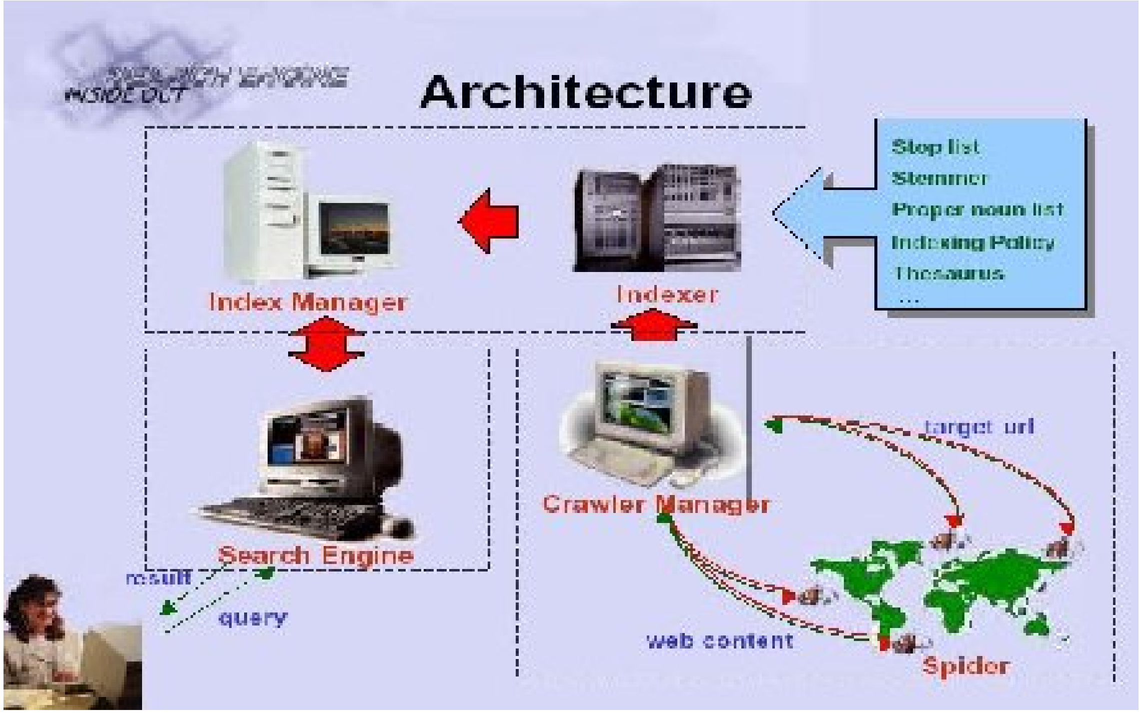


Figure 4.1 : Search Engine Architecture

4.2.2 Indexing

The index manager is at the heart of any Search Engine. It takes all the documents that have been processed by the Crawler, and for each document it extracts a set of terms that will be used to aid in the retrieval of that document at a later stage. These terms, often referred to as *Index Terms*, provide a logical view of the document Baeza Yates et al (1999). Traditionally, index terms would be manually defined and would only represent a small subset of the total document content. However, advances in computing are now making it possible to represent a document by its full set of keywords. Baeza Yates et al (1999) refers to this as a full text logical view. However, even though modern computer systems have the capacity to handle large sets of index terms it is still sometimes necessary to reduce these for efficiency purposes. This is normally achieved by applying a set of intelligent text operations to the text. These text operations attempt to create a set of index terms that make best use of the resources available and that accurately match potential user query terms www.csee.umbc (2008). The following table 4.1 shows some of the text operations that can be applied G.Saltonl (1989).

Table 4.1 : Text Operations

Text Operation	Description
Lexical Analysis	Conversion of a byte stream into a set of tokens.
Elimination of Stopwords	Removes words that tend not to enforce semantic content of the document. E.g. the, of, and, a, in, to, is for, with, are, etc…
Stemming	Reduction of terms to a single stem form by removing suffixes and prefixes such as –‘s, -ing, -ed, -s, … in-, ad-, pre-, sub,…
Building a thesaurus	Build a thesaurus for alternatives to the terms identified after applying above text operations.

In Chapter 5 some of these operations will be covered in detail.

Once the set of index terms has been properly identified the process of indexing the documents is carried out. In many Search Engines this is achieved by using a technique known as ***inverted file indexing***. Basically, an inverted file is a list of sorted index terms where each term maintains a set of pointers to the documents in which it occurs. However, at this point it is worth noting that in addition to the complexities of creating the initial inverted file, one major problem associated with this technique is in the maintenance of the index file itself. Quite often, it is necessary to add, remove or modify the terms in the index when new documents become available or when existing documents are removed from the document set. Traditionally, this update would be achieved by the index manager creating a new inverted file and destroying the old one. This technique can be quite time consuming especially when the document set being indexed is large. In the research a fixed inverted file technique will be used, because of using desktop application with fixed already crawled documents Huang.L (2008).

4.2.3 Search Interface

The search interface is the visible part of a Search Engines Darnell.R (2008).

The search interface can be divided into three distinct areas, two of which provide the user interface. These are the:

- Query Interface
- Ranking software
- Results Interface

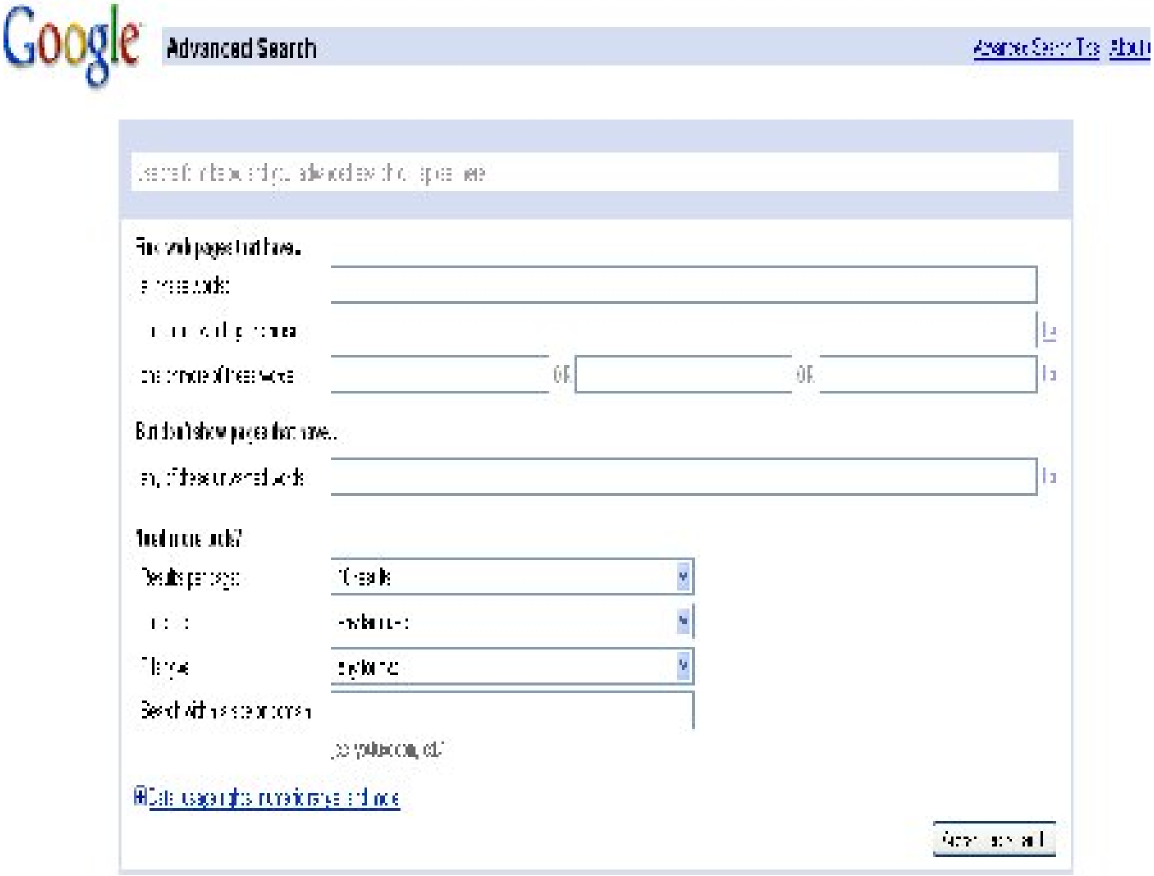
4.2.3.1 Query Interface

The query interface provides the users with an interface that will allow them to submit a query to the underlying Search Engine. In general this interface simply presents the user with a simple edit box into which a set of words can be typed; this is often referred to as a keyword search.

It is understandable that a user would expect that any given set of keywords would result in the same query in all Search Engines. However, this is not always the case, as Baeza Yates et al (1999) points out, in AltaVista Search Engine (2008) a sequence of words is a reference to the union of all the Web pages having at least one of those words, while in HotBot it is a reference to the Web pages having all the words. In addition to simple keyword searches, most Search Engines will also provide a query interface for complex queries that require techniques such as Boolean operations, phrase searches, proximity searches etc.

This is illustrated in Figure 4.2 which shows the advanced options provided by Google Search Engine.

Google's advanced search provides a comprehensive range of filtering techniques including phrase matching, word exclusion, file format selection and even the restriction of a search to particular domains.



Google
Advanced Search
Advanced Search Tips | About
Use the form below and your advanced search will appear here
Find web pages that have...
all these words:
this exact wording or phrase:
one or more of these words:
OR
OR
But don't show pages that have...
any of these unwanted words:
Need more tools?
Results per page:
10 results
Language:
any language
File type:
any format
Search within a site or domain:
(e.g. youtube.com, .edu)
⊞Date, usage rights, numeric range, and more
Advanced Search

Figure 4.2 : Google advanced Search

As an alternative to the typical query interface outlined above, other Search Engines such as **Yahoo**, **LookSmart** and **Magellan** use a technique known as the search directory technique. A search directory allows hierarchical searches, starting with general subject headings and moving down to increasingly more specific subheadings.

The searching is done through the use of fixed menu categories, arranged in a hierarchical fashion. The user simply clicks on a category of interest and is presented with all of the sub-categories contained within. The user continues this navigational approach until satisfied with the results.

This search technique is considered to be quite good for scenarios where the user is unsure of which keywords to use, however, it should be noted that if the search directory allows the people submitting the content to chose their own categories, rather than using reviewers to categorise site submissions, then the quality of the system database can be affected, and the data content may be placed in unsuitable categories.

4.2.3.2 Ranking Software

The ranking software is the component of a Search Engine that turns a user request into a set of retrieved documents. As Marendy.P (2001) points out, this ***software is responsible for sifting through the data in the indexes to find pages that are deemed relevant, and producing a ranked output of those pages*** Marendy.P (2001). There are a number of techniques that have been developed to rank retrieved documents. They fall into two categories, those based on classical information retrieval techniques and those specifically developed for the Web.

The two most popular types of ranking algorithms that are used today are based on the classical Boolean and Vector State models.

The Boolean model is based on algebraic set theory. For a given user query such as [‘information’ **AND** ‘retrieval’] the documents within the set are examined to determine which ones share the same terms as the query. If any document shares the same terms then it is included in the result set otherwise it is not. In its most absolute form the Boolean model insists that exact matches are found and gives no precedence to documents which have a greater frequency of the query term occurrences.

The vector state model, on the other hand, takes a completely different approach to relevancy and ranking. In this model each document is modelled in software as a vector, each coordinate relating to a particular attribute of the document. When a query is submitted a vector is produced for it and a ranking for each document is determined based on each document's distance to the query Kobayashi.M et al (2000).

A traditional method for determining this distance is based on the computation of the angle defined by the query and document vectors. Although becoming popular the vector state model is considered impractical for large databases such as the World Wide Web.

With the Web becoming ever more popular as an information retrieval system, certain researchers have proposed information retrieval techniques that incorporate the use of the structural components of the Web-pages themselves. Carriere and Kazman (1997) propose a model built around the connectivity of pages. After a keyword query has been submitted, a set of results known as the hit set is returned. This is converted into a neighbour set, which contains for each node in the hit set, all the nodes that it links to or that link to it.

The connectivity of a page can then be determined by the total number of incoming and outgoing hyperlinks to that page. The pages are ranked, giving the highest rank to the most highly connected pages.

Carriere and Kazman (1997) have found that by doing this they are able to pinpoint hot spots that are particularly relevant to the user's query. In addition they have also extended their model to search beyond the returned result set to locate interesting pages that are highly connected to those returned by the original query. Google which indexes over 8 billion Web pages is just one example of a commercial Search Engine using such technology Kobayashi.M et al (2000).

Lawrence and Sergey Brin (1998) founded instead of focusing on content-based for ranking the hits it focuses on ranking the documents off-line of user through the inner and outer links that's associated with each document Brin. S. et al (1998).

Assume that page A has pages T_1, T_2,…,T_n which point to it. And parameter d is a damping factor which can be set between 0 and 1 (often d assigned at 0.85). Also, C(A) is considered as the number of links going at of page A. the Page Rank of a page A is as following :

$$\boldsymbol{PR(A)} = (1 - \boldsymbol{d}) + \boldsymbol{d} \times \left(\boldsymbol{PR(T1)} / \boldsymbol{C(T1)} + \cdots \boldsymbol{PR(Ti)} / \boldsymbol{C(Ti)} \cdots + \boldsymbol{PR(Tn)} / \boldsymbol{C(Tn)} \right) \text{......} \; 4.1$$

Brin. S. et al (1998).
where:

PR(A): is the Page Rank of page A.
PR(Ti): is the Page Rank of page Ti which is a link to page A.
C(Ti) : is the number of outbound links on page Ti.
d : is a damping factor which can be set between 0 and 1.

It is obvious that Page Rank dose not rank Web sites as a whole, but is determined for each page individually. Further, the Page Rank of page A is recursively defined to be the Page Rank of those pages which link to page A . By recursion, Page Rank computations repeated several times until the number stop changing much. This can be considered as iteration or recursion stopping condition.

4.2.3.3 Results Interface

The human-computer interface is one aspect of the Search Engine technology that is less well understood. There are a number of reasons for this, but in part it is because humans are complex creatures and they tend to be more difficult to characterize and measure than standard computer systems. It is also because Search Engines often return extremely large result sets of data which, includes relevant results, as well as many irrelevant ones.

This provides complications for the users in their filtering process. Studies in this area have lead to a growing technology called information visualization, which attempts to provide visual depictions of very large information spaces in a way that benefits the user filtering process. Traditional Search Engines such as Google and Yahoo Search Engine (2008) tended to return lists of matched Web pages, known as hit lists shown in Figure 4.3. Often these results are accompanied by a score that is intended to indicate the relevance of each node.

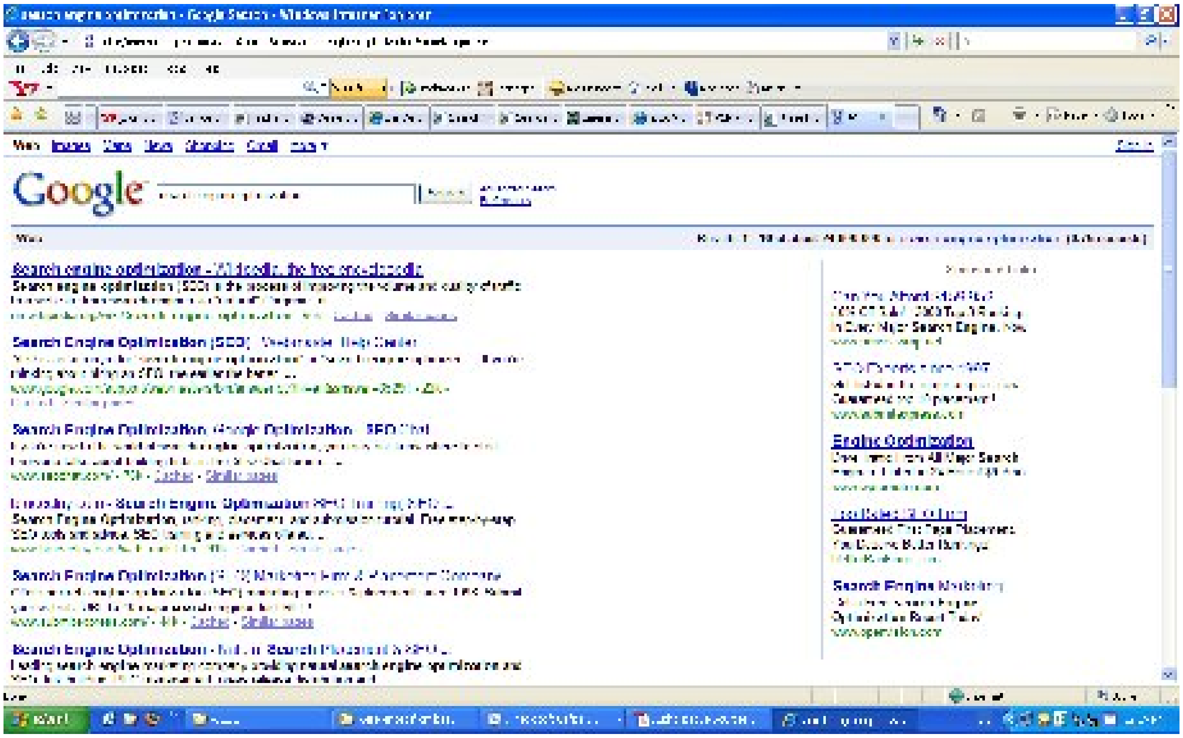

Web Images Maps News Shopping Gmail more
Google
Web
Search engine optimization - Wikipedia, the free encyclopedia
Search Engine Optimization (SEO) - Webmaster Help Center
Search Engine Optimization, Google Optimization - SEO Chat
Search Engine Optimization (SEO) Marketing
Search Engine Optimization
Engine Optimization
Search Engine Marketing
start

Figure 4.3: Google hit list (results)

However, this method possesses a number of problems. ***Firstly***, as already mentioned, the result set may often contain results that are of no relevance to the user. ***Secondly***, if the result sets are particularly large the user may ignore results that have been ranked at the lower end of the list but may be relevant to their search. Because users don’t necessarily know what ranking mechanism the Search Engine is using to calculate the score they can’t be sure if the score produced is an accurate reflection of the result relevance.

For example, a user that submits a query [‘hypertext’ **AND** ‘indexing’ **AND** ‘retrieval’] could be presented with a hit list where documents with all the keywords are given the highest scoring. However, the user may be interested in results that have a higher proportion of the term ‘indexing’. Unfortunately, with traditional systems, they would have to review the documents before obtaining a clear indication as to their relevance. In the following text the attempt will be to outline a number of alternative visualization techniques:

A. One alternative as outlined by Liu.B et al (2002) is to use ***hierarchical clustering***. Hierarchical clustering produces a nested sequence of clusters which form a tree like structure. The bottom nodes of the tree represent single Web pages, these are combined so as pages that are most similar are clustered together under a single node. The process continues up each level until all pages are merged into a single cluster known as the root cluster.

In the schematic example in Figure 4.4 we have 5 Web pages represented as clusters at the bottom of the tree. Cluster 6 is formed by the merging of clusters 1 and 2. Clusters 3 and 6 are then merged to form cluster 8 and so on until all pages are clustered under the topmost node which is cluster 9.

With each node valuable pieces of summary information such as keywords and domain information are also stored to aid the user. Using the tree the user can drill down and roll up to see pages that are contained at any level of granularity. At any stage, the user can ‘click’ on a cluster to receive the summary information. Clustering the pages this way takes advantage of the superb visual capability of human users to enable them to spot interesting patterns, pages and information easily Liu.B et al (2002).

B. The NIRVE prototype as illustrated in Figure 4.5. It takes the idea of clustering one step further. Rather than simply allowing the Search Engine to cluster pages based on its internal ranking algorithm the NIRVE prototype allows users to dynamically map a subset of keywords of the query into a concept. Names and colours can be assigned to these concepts which are displayed as an interactive legend at the bottom of the document space Sebrechts.M.M et al (1999).

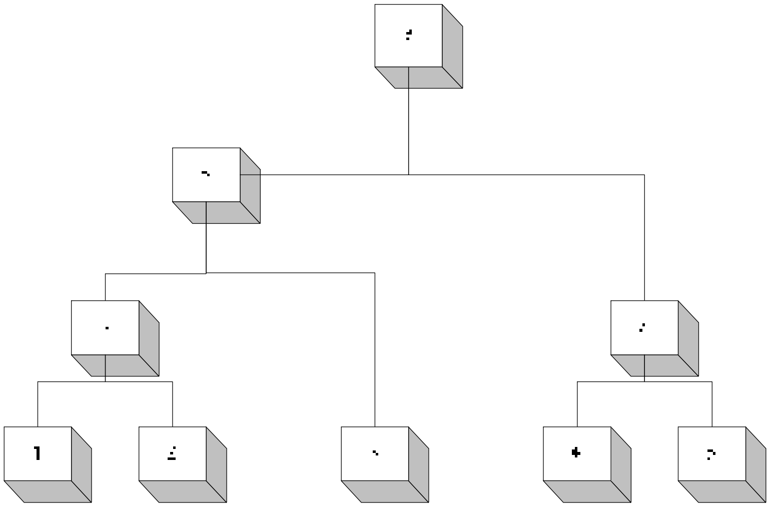


Figure 4.4: Hierarchical Clustering

For each retrieved document a concept profile (keywords, word frequencies) produced based on the frequency of concept keywords that are contained within the document. Clusters are then defined based on the sets of documents that have the same concepts associated with them. Instead of simply presenting these concepts in a tree form as before the NIRVE prototype represents each cluster as a small box on the face of a globe.

The cluster box contains a colour bar chart indicating the average concept profile of the documents it contains. The thickness of the cluster box is proportional to the number of documents within the cluster. The latitude of the cluster icon on the globe is determined by the number of concepts exemplified by the cluster i.e. the more concepts a cluster contains the further ‘north’ it will be located. In addition to this, clusters that differ by a single concept are connected by an arc whose colour represents the conceptual difference between them.

Selecting any cluster on the globe causes a 2D-document rectangle to be opened which contains a list of all documents within the cluster. Although users of the NIRVE prototype felt that the visualization techniques were quite intuitive it was found that as more concepts were added and the number of clusters increased it became more difficult to use the system. The reason for this was that, due to the increased number of clusters their size on screen for a given view decreased, leading to difficulty distinguishing the concept colours. In addition, once more than eight concepts were defined it became difficult to read concept labels and keywords in the legend.

C. Hearst outlines a more compact form of visualization display through the use of the TileBars interface. TileBars are graphical bars that are displayed next to the title of retrieved documents. They attempt to illustrate to the user both the sections of the documents that contain the keywords as well as the frequency with which these keywords appear in the document Hearst.M.A (1995).

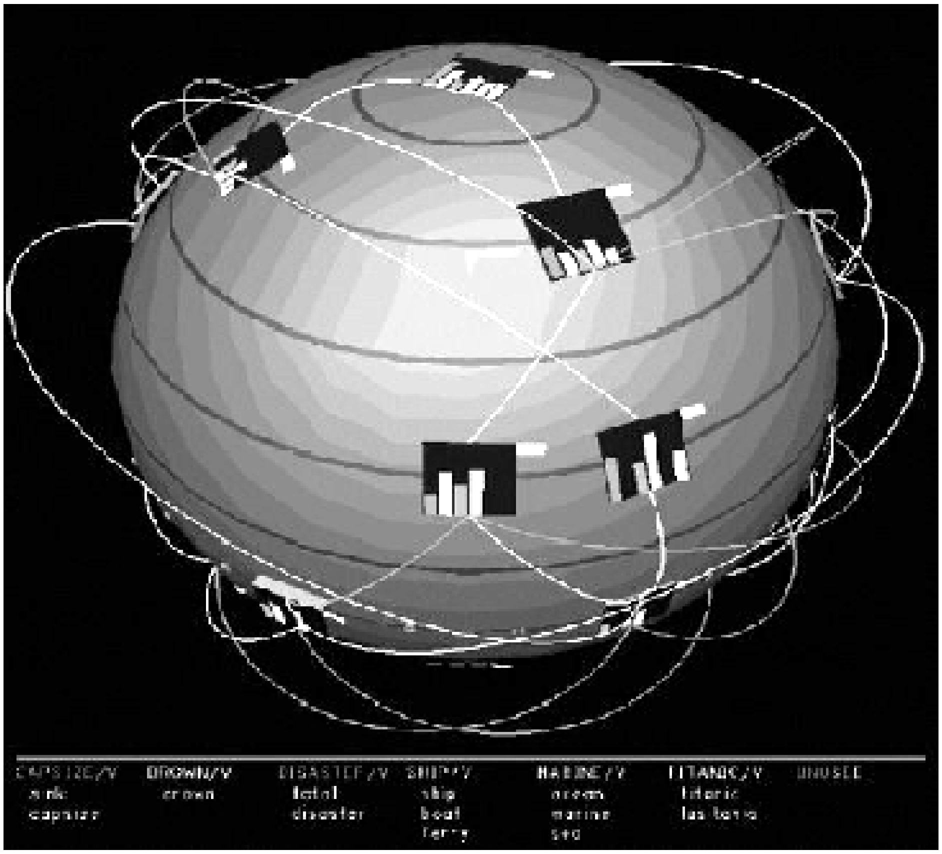
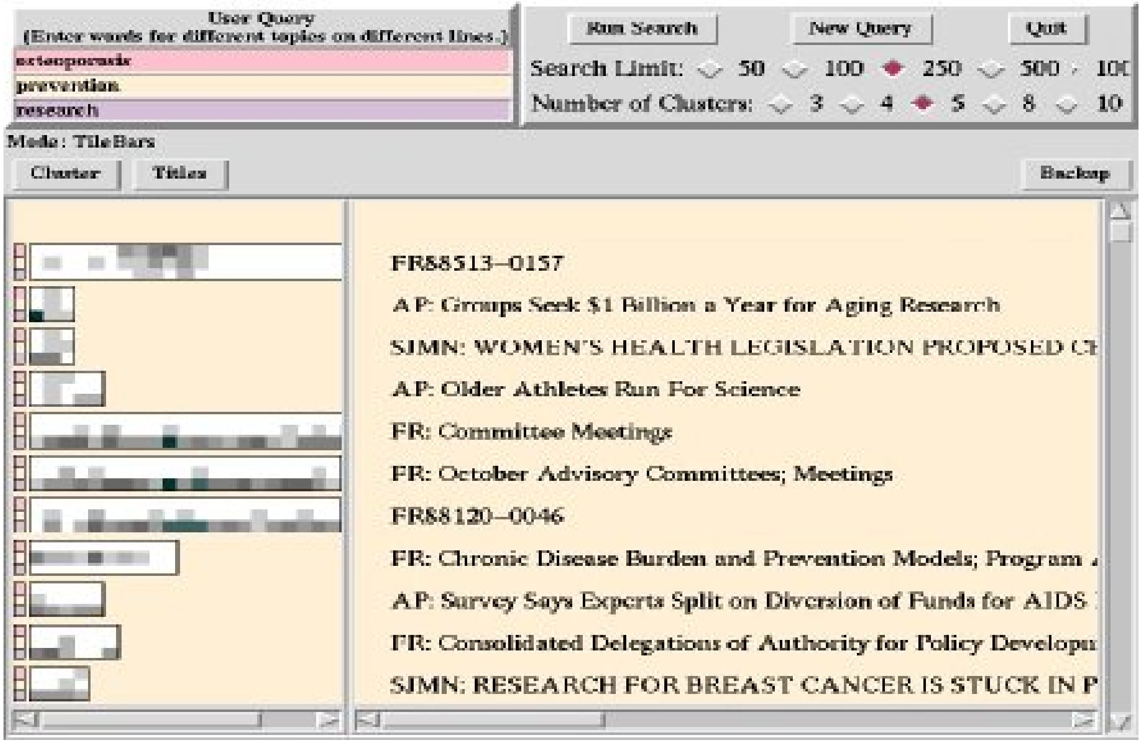


Figure 4.5: NIRVE Prototype

The bars are subdivided into columns where each column represents a section within the text. Each column is further subdivided into coloured rows, where each row and colour represents a keyword within the query. Thus it ends up with a bar comprised of a set of coloured squares. The darkness of each square is intended to correspond to the frequency of the query term within the text segment, the darker the square the more occurrences within the text. White indicates no match at all. This method aims to provide the user with an overview of the document at a glance.

Figure 4.6 provides an example where the query indicated the retrieval of documents with the terms [‘osteoporosis’ **OR** ‘prevention’ **OR** ‘research’].



osteoporosis
prevention
research

Search Limit: ◇ 50 ◇ 100 ◆ 250 ◇ 500 › 10(

Number of Clusters: ◇ 3 ◇ 4 ◆ 5 ◇ 8 ◇ 10

Mode: TileBars

FR88513–0157
AP: Groups Seek $1 Billion a Year for Aging Research
SJMN: WOMEN'S HEALTH LEGISLATION PROPOSED CH
AP: Older Athletes Run For Science
FR: Committee Meetings
FR: October Advisory Committees; Meetings
FR88120–0046
FR: Chronic Disease Burden and Prevention Models; Program A
AP: Survey Says Experts Split on Diversion of Funds for AIDS I
FR: Consolidated Delegations of Authority for Policy Developm
SJMN: RESEARCH FOR BREAST CANCER IS STUCK IN P

Figure 4.6: Tilebar Visualization

The visualisation techniques outlined above are just a few of the many being researched today. Others include the ‘Bullseye’ view for connectivity based retrievals proposed by Carriere and Kazaman (19997) as well as simple 2D grid models as outlined by Sebrechts.M.M et al (1999).

4.3 Google Search Engine As a Prototype

The amount of the information on the Web is growing rapidly, as well as the number of new user inexperienced in the art of Web research. So, it becomes necessary to create a Search Engine which scale even today’s presents many challenges.

Such a Search Engine must have a fast crawling technology to gather the Web documents and keep them up to date. Storage space must be used efficiently to store indices and optionally the documents themselves. In the needed Search Engine the indexing system must process hundreds of gigabytes of data efficiently, and the queries must be handled quickly, at a rate of hundreds to thousands per second.

The word Google comes from the word googol, which means 10^{100} . The Google Search Engine (www.Google.com) heavily uses the structure present in hypertext to provide much quality search results.

Google does not require nor request submission to their site. They don't ask for payment to be listed. They have a technology called spider that crawls the Internet looking for content to list. This includes Web pages, Adobe Acrobat (PDF) files, images, news, and Usenet discussion groups. The cost of indexing and storing text or Hypertext Markup Language (**HTML**) will eventually decline relative to the amount that will be available Brin. S. et al (1998).

4.3.1 System Features

The Google Search Engine has two important features that help it produce high precision results. First, it makes use of the link structure of the Web to calculate a quality ranking for each Web page. This ranking is called PageRank. Second, Google utilizes link to improve search results.

4.3.1.1: Page Rank

Google ranking algorithm is based on how each Web page is connected. The page rank algorithm uses equation 4.1.

The Google Search Engine uses a simple algorithm to calculate page rank. It simulates users using the Search Engine and applies the equation to rank the Web pages. It allows rapid calculation: about 26 million Web pages can be computed in a few hours.

4.3.1.2 Anchor Text

Most Search Engines associate the text of a link with the page that the link is on. However, Google associates it with the page that link points to it. There are several advantages for the latter approach. First, anchors often provide more accurate descriptions of Web pages than the pages themselves. Second, anchors may exist for documents which cannot be indexed by a text-based Search Engine, such as images, programs, and databases. So the Search Engine would not return non-existing pages to user. Google has over 259 million anchors indexed for their crawl of 24 million pages. The idea of anchor text was originated by World Wide Web Worm (WWWW), Hearst.M.A (1995), Oliver.A.McBryan (1994).

4.3.2 Google System Anatomy

First, a high level discussion of architecture will be provided. Then, there is some in-depth description of information data structures. Finally, the major applications: crawling, indexing, and searching.

4.3.2.1 Google Architecture Overview

Most of Google is implemented in **C** or **C++**, and can run in either Solaris or Linux. The whole system work is illustrated in Figure 4.7.

In Google, the URL Server sends lists of URLs to be fetched by the crawlers. The crawlers download pages according to the list and send the downloaded pages to the Store Server. This is done by several distributed crawlers. The Store Server compresses the pages and stores them in the repository. Every Web page has an associated ID number called a docID, which is assigned whenever a new URL is parsed out of a Web page.

The indexing function performed by the indexer and the sorter. The indexer performs a number of functions. It reads the repository, uncompresses the documents, and parses them. Each document is converted into a set of word occurrences called hits.

The hits contain information about a word, position in document, an approximation of font size, and capitalization. The indexer distributes these hits into a set of “barrels” and creates a partially sorted forward index.

The indexer performs another important function. It parses out all the links in every Web page and stores important information about them in an anchors file. This file contains enough information to determine where each link points from and to, and the text of the link. After that, the URL Resolver reads the anchors file and converts relative URLs into absolute URLs and in turn into docID (as an example: http://www.google.com/ ⟹ 209.85.129.104 ⟹ 1950). It puts the anchor text into the forward index, associated with the docID that the anchor points to. It also generates a database of links, which are pairs of docIDs, for storing links and docIDs. The database is used to compute the PageRank for all the documents.

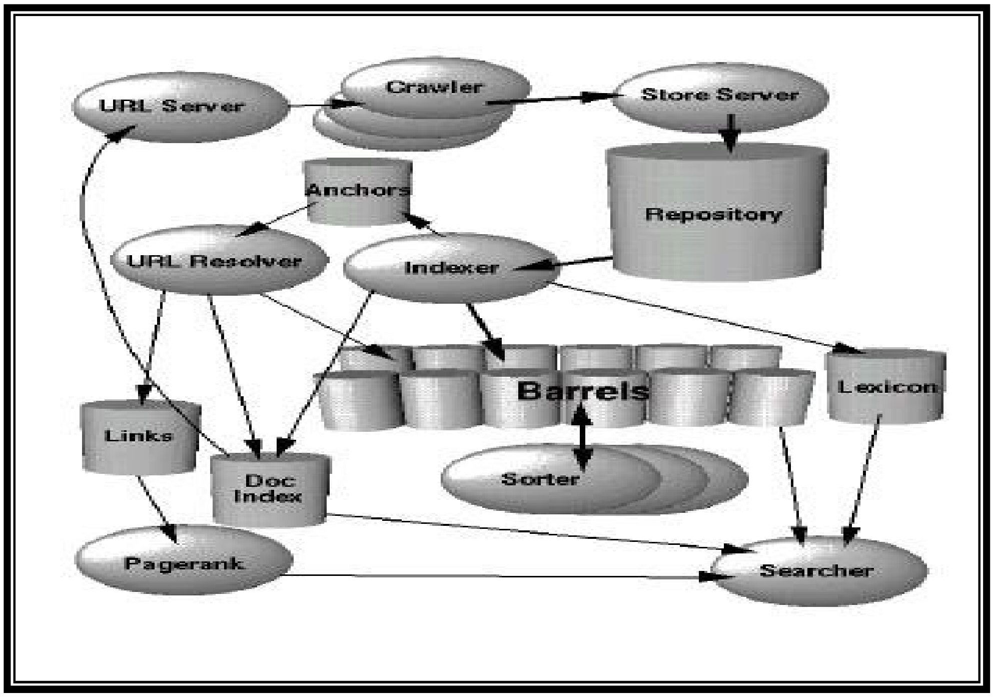


Figure 4.7. The architecture of Googol Search Engine

The Sorter takes the barrels and resorts them by wordID instead of docID in order to generate the inverted index. Also, the Sorter produces a list of wordIDs and offsets into the inverted index. A program called DumpLexicon takes this list together with the lexicon produced by the indexer and generates a new lexicon to be used by the searcher. The searcher is run by a Web server and uses the lexicon built by DumpLexicon together with the inverted index and the PageRanks to answer queries Brin. S. et al (1998).

4.3.2.2 Google Data Structure

Google's data structures are optimized so that a large document collection can be crawled, indexed, and searched with little cost. Although, CPUs and bulk input output rates have improved dramatically over the years, a disk seek still requires about 10 ms to complete. Google is designed to avoid disk seeks whenever possible, and this has had a considerable influence on the design of the data structures.

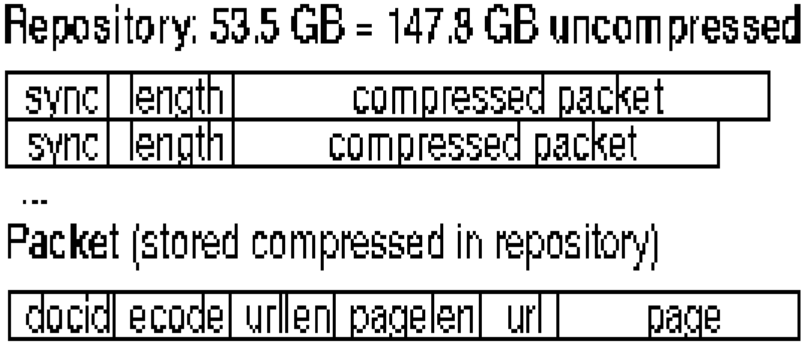


Figure 4.8 Repository Data Structure

Google system is designed from virtual files spanning called BigFiles, repository which contains the full HTML for every Web page as Figure 4.8 shows, document index which keeps information about each document it is fixed width Index sequential access mode (ISAM) index, ordered by docID, lexicon, hit list corresponds to a list of occurrences of a particular word in a particular document including position, font, and capitalization information, forward index which is actually already partially sorted; it is stored in a number of barrels (used 64). Each barrel holds a range of wordID's, and inverted index which consists of the same barrels as the forward index, except that they have been processed by the sorter as shown in Figure 4.9.

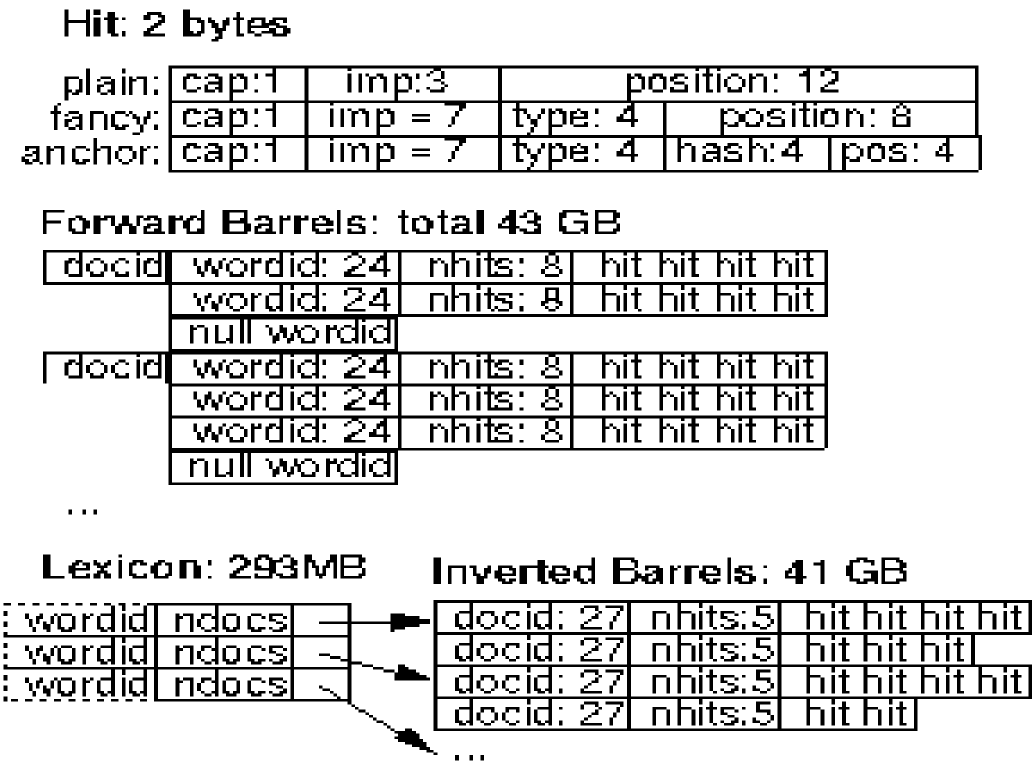

Hit: 2 bytes
plain: cap:1 imp:3 position: 12
fancy: cap:1 imp = 7 type: 4 position: 8
anchor: cap:1 imp = 7 type: 4 hash:4 pos: 4
Forward Barrels: total 43 GB
docid wordid: 24 nhits: 8 hit hit hit hit
wordid: 24 nhits: 8 hit hit hit hit
null wordid
docid wordid: 24 nhits: 8 hit hit hit hit
wordid: 24 nhits: 8 hit hit hit hit
wordid: 24 nhits: 8 hit hit hit hit
null wordid
...
Lexicon: 293MB
wordid ndocs
wordid ndocs
wordid ndocs
Inverted Barrels: 41 GB
docid: 27 nhits:5 hit hit hit hit
docid: 27 nhits:5 hit hit hit
docid: 27 nhits:5 hit hit hit hit
docid: 27 nhits:5 hit hit
...

Figure 4.9 Forward and Reverse Indexes and the Lexicon

Chapter Five

Design and Implementation Of the Proposed System

5.1 Introduction

The Internet forms a distributed library of billion of pages, one that is accessible to anyone, anywhere in the world, at the click of the mouse. Every day hundreds of millions of “trips” to the library start with a query to an Internet Search Engine. For example if the query is “library of the congress” Google Search Engine will return in less than a second a list of 38,900,000 pages, stored by their usefulness to the query. This unprecedented ease of access to information has revolutionized the way research is done by students, scientists, journalists, shoppers, and others. It opens up an online marketplace of products, services, and ideas that benefits both information providers and seekers; sellers and buyers; consumers and advertisers.

The design of a good Search Engine confronts many competitions. In particular, the Search Engine must deal with huge volumes of data. Unless it has unlimited computing resources and unlimited time, one must carefully decide what Web pages to retrieve and in what order. Nowadays, the most familiar Search Engine is *Google Search Engine* that is explained explicitly in the previous chapter (section 4.3).

In this chapter, several enhancements are proposed by constructing a fresh Search Engine. The goal from these propositions is to retrieve a well ranked result. The goal will be achieved by implementing the following issues:

1. Enhancing indexing process by extracting several factors. The factors are used to provide more accurate information about the content of the document.
2. Implementing a new ranking module used for computing the rank score according to the Global Ranking (ranking that depends on the hyperlink structure and features among Web pages) and Local Ranking (ranking that depends on the relevancy between the given query and Web pages).

5.2 Proposed System Architecture

The proposed system is constructed in a combined link-based and content-based response fashion, as it is shown in Figure 5.1. The search starts when the user posts a query to the Search Engine interface. The query is forwarded to the query optimizer and then once it is constructed, the query is posted to the searcher, the searcher performs several operations to retrieve the required documents from the repository, and then the documents are ranked by the ranker. Finally, the results are browsed to the user as a sorted list of pages' title and their URL according to their accurate ranking score.

The system is partitioned into two parts depending on connectivity with the user. These parts are off-line part, and on-line part. Each part of the system will be described in detail as follows:

5.2.1 Off-Line Part

This part is run independently from the end-user. It consists of the following sub-systems and databases:

A. Crawler

A crawler is a program that collects Web pages, commonly for use by Search Engine Brian Pinkerton (1994). The basic algorithm executed by any Web crawler takes a list of URLs as its input and repeatedly executes the following steps. Remove a URL from the URLs list, download the corresponding document, and extract any links contained in it. For each of the extracted links, ensure that it is an absolute URL, add it to the list of URLs, and repeat the process again. This basic algorithm requires a number of functional components:

1. A component to store the list of URLs.
2. A component for downloading documents using the HTTP protocol.
3. A component for extracting links from HTML documents.
4. A component for determining whether a URL has been encountered before.

The main question here is: How should a crawler select URLs to scan from its queue of known URLs? If a crawler intends to perform a single scan of the entire Web, and the load placed on target sites is not an issue, then any URL order will suffice. That is, eventually every single known URL will be visited, so the order is not critical.

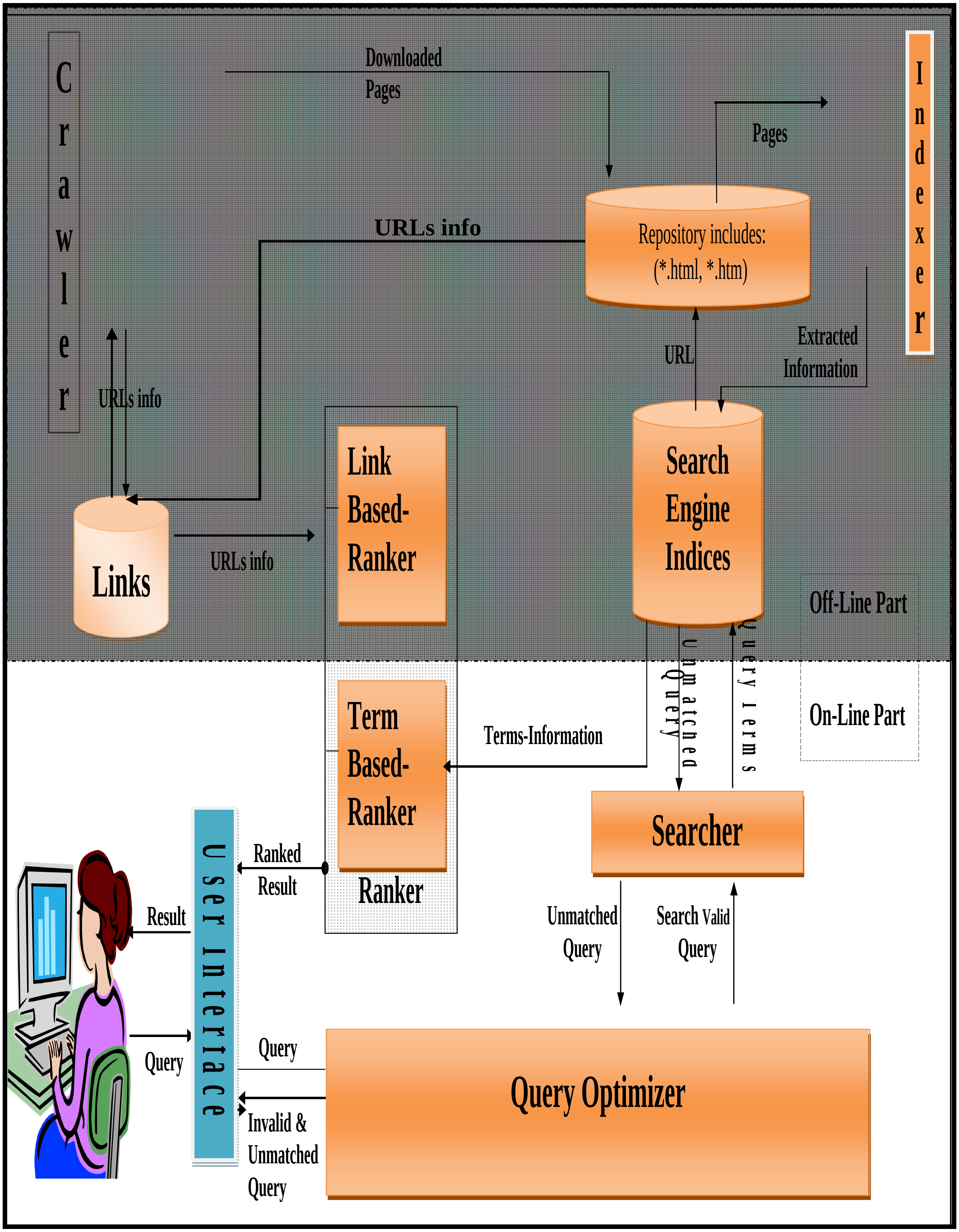


Figure 5.1: The Proposed Search Engine Diagram

However, most crawlers will not be able to visit every possible page for two main reasons:

1. The crawler may have limited storage capacity, and may be unable to index or analyze all pages. Currently, the Web contains several terabytes of data and is growing rapidly, so it is reasonable to expect that most systems will not want or will not be able to cope with all that data Brewster Kahle (1997).
2. Crawling takes time, so at some point the crawler may need to start revisiting previously scanned pages, to check for changes. This means that it may never get to some pages. It is currently estimated that over 600GB of the Web changes every month Brewster Kahle (1997).

In either case, it is important for the crawler to visit "important" pages first, so that the fraction of the Web that is visited is more meaningful. The following sections will present several different useful definitions of importance, and develop crawling priorities so that important pages have a higher probability of being visited first.

Several importance measures were investigated to establish site importance given a Web page P, the importance of the page *I (P)* will be defined in one of the following ways:

1. **Similarity to a Driving Query** *Q*. A query Q drives the crawling process, and I(P) is defined to be the textual similarity between P and Q. IS(P) will be used to refer to the importance metric in this case. Also IS (P, Q) refers to make the query explicit.

To compute similarities, each document (P or Q) can be viewed as an n-dimensional vector ($w_1, . . . ,w_n$). The term w_i in this vector represents the i^{th} word in the vocabulary. If w_i does not appear in the document, then w_i is zero. If it does appear, w_i is set to represent the significance of the word. One common way to compute the significance w_i is to multiply the number of times the i^{th} word appears in the document by the inverse document frequency (*idf*) of the i^{th} word. The *idf* factor is one divided by the number of times the word appears in the entire "collection," which in this case would be the entire Web. The *idf* factor corresponds to the content discriminating power of a word: A term that appears rarely in documents (e.g., "queue") has a high *idf*, while a term that occurs in many documents (e.g., "the") has a low *idf*. The similarity between P and Q can then be defined as the inner product of the P and Q vectors. Another option is to use the cosine similarity measure, which is the inner product of the normalized vectors. Note that if *idf* terms haven't been used in the similarity computation, the importance of a page, IS (P), can be computed with "local" information, i.e., just P and Q. However, by using *idf* terms, it can be computed with "global" information. During the crawling process the IS` (P) is used to refer to the estimated importance of page P, which is different from the actual importance IS (p) that is computable only after the entire Web has been crawled. If *idf* factors are not used, then IS` (p) = IS (p) Brewster Kahle (1997), Budi Yuwono et al (1995). The similarity based algorithm is given in Algorithm 5.1.

Algorithm 5.1   Crawling algorithm (similarity based)

// URL : Unified recourse locator.
// hot_queue: Queue that holds the URLs which contains search words.
// url_queue : Queue that holds URLs.
Input: starting_url: seed URL
Procedure:

```
enqueue(url_queue, starting_url)
while (not empty(hot_queue) and not empty(url_queue))
    url = dequeue2(hot_queue, url_queue)
    page = crawl page(url)
    enqueue(crawled_pages, (url, page))
    url_list = extract_urls(page)
    foreach u in url_list
        enqueue(links, (url, u))
        if (u∉url_queue and u∉hot_queue and (u,-) ∉crawled_pages)
            {if (u contains computer in anchor or url)
                enqueue(hot_queue, u)
            else
                enqueue(url_queue, u)}
    reorder_queue(url_queue)
    reorder_queue(hot_queue)
end while
```

Function description:

```
dequeue2(queue1, queue2): if (not empty(queue1)) dequeue(queue1)
                          else dequeue(queue2)
```

2. Backlink Count*:* The value of *I* (*P*) is the number of links to *P* that appear over the entire Web. *IB* (*P*) will be used to refer to this importance metric. Intuitively, a page *P* that is linked to by many pages is more important than one that is seldom referenced. This type of “citation count” has been used extensively to evaluate the impact of published pages. On the Web, *IB* (*P*) is useful for ranking query results, giving end-users pages that are more likely to be of general interest. Note that evaluating *IB* (*P*) requires counting backlinks over the entire Web. A crawler may estimate this value with *IB*` (*P*), the number of links to *P* that have been seen so far Budi Yuwono et al (1995), Junghoo Cho et al (1998).

3. PageRank*:* The *IB* (*P*) metric treats all links equally. Thus, a link from the Yahoo home page counts the same as a link from some individual’s home page. However, since the Yahoo home page is more important (it has a much higher *IB* count), it would make sense to value that link more highly. The PageRank backlink metric, *IR* (*P*), recursively defines the importance of a page to be the weighted sum of the importance of the pages that have backlinks to *P*. Such a metric has been found to be very useful in ranking results of user queries. *IR*` (*P*) will be used for the estimated value of *IR* (*P*) when there is only a subset of pages available.

More formally, if a page has no outgoing link, assume that it has outgoing links to every single Web page. Next, consider a page *P* that is pointed at by pages $t_1, \ldots, t_n$. Let c_i be the number of links going out of page t_i. Also, let *d* be a damping factor (whose intuition is given below). Then, the weighted backlink count of page *P* is given by:

$$IR\ (P) = (1 - d) + d\ [IR\ (t_1)/c_1 + \cdot\ ^{\cdot}\ IR(t_i/c_i)\cdot\ \cdot + IR(t_n)/c_n]\ \ldots\ldots\ (5.1)$$

This leads to one equation per Web page, with an equal number of unknowns. The equations can be solved for the *IR* values. They can be solved iteratively, starting with all *IR* values equal to 1. At each step, the new *IR* (*p*) value is computed from the old *IR* (t_i) values (using the equation above), until the values converge. Algorithm 5.2 shows the link based algorithm Brin.S et al (1998).

As an example of how to find IR(P) for a couple of pages Figure 5.2 demonstrates that

Suppose the damping factor = 0.85.

IR(A) = 0.15 + 0.85 IR(C)

IR(B) = 0.15 + 0.85 (IR(A) / 2)

IR(C) = 0.15 + 0.85 (IR(A) / 2 + IR(B))

These equations can be solved by assuming at first IR(A) , IR(B) , IR(B) to be 1.

IR(A) = 1.193633507

IR(B) = 0.657294240

IR(C) = 1.215994345

The page rank [IR(P)] results after 10 iterations.

4. ***Forward Link Count:*** For completeness it may considers a metric *IF* (*P*) that counts the number of links that emanate from *P*. Under this metric, a page with many outgoing links is very valuable, since it may be a Web directory. This metric can be computed directly from *P*, so *IF*` (*P*) = *IF* (*P*). This kind of metric has been used in conjunction with other factors to reasonably identify index pages.

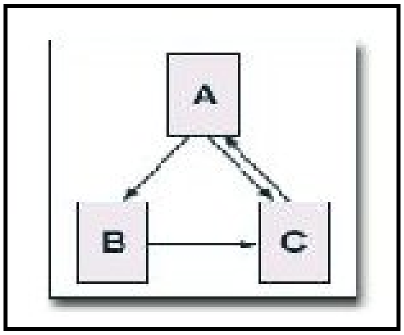


Figure 5.2: Example of small Web

5. ***Location Metric:*** The *IL* (*P*) importance of page *P* is a function of its location, not of its contents. If URL *u* leads to *P* then *IL* (*P*) is a function of *u*. For example, URLs ending with “.com” may be deemed more useful than URLs with other endings, or URLs containing the string “home” may be more of interest than other URLs. Another location metric that is sometimes used considers URLs with fewer slashes more useful than those with more slashes. All these examples are local metrics since they can be evaluated simply by looking at the URL *u*. As stated earlier, the importance metrics can be combined in various ways. For example, a metric may defined as:

$$IC\,(P) = k_1 \cdot IS\,(P,Q) + k_2 \cdot IB(P) \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ 5.2$$

For some constants k_1, k_2. This combines the similarity metric (under some given query Q) and the back-link metric. Pages that have relevant content and many back-links would be the highest ranked Junghoo Cho et al (1998).

Two main aspects could be devised. In spidering algorithms which consider only *PageRank* and *Backlink count*, the PageRank strategy outperforms the other due to its non–uniform traversing behaviour: going in Depth when the importance of the children

Algorithm 5.2 Crawling algorithm (link based)

// URL : Unified recourse locator.
// hot_queue: Queue that holds the URLs which contains search words.
// url_queue : Queue that hold URLs.

Input: starting_url: seed URL

Procedure:

```
enqueue(url_queue, starting_url)
while (not empty(url_queue))
    url = dequeue(url_queue)
    page = crawl_page(url)
    enqueue(crawled_pages, (url, page))
    url_list = extract_urls(page)
    for each u in url_list
       enqueue(links, (url, u))
       if (u$\notin$url_queue and (u,-)$\notin$crawled_pages)
         enqueue(url_queue, u)
reorder_queue(url_queue)
end while
```

Function description:

enqueue(queue, element):   append element at the end of queue
dequeue(queue) : remove the element at the beginning of queue and return it
reorder_queue(queue) : reorder_queue using information in links (refer to Algorithm 5.3)

is high enough, moving to the siblings whenever children nodes contain unimportant documents.

On the other hand, when a similarity driven crawling algorithm is used the PageRank strategy is comparable to Breadth First traversal. This happens because when a page is authoritative with respect to a particular topic, its children are likely to have a high importance too Budi Yuwono et al (1995), Junghoo Cho et al (1998).

Running a Web crawler is a challenging task. There are complex performance and reliability issues. Crawling is the most fragile application since it involves interacting with hundreds of thousands of Web servers and various server names which are all beyond the control of the system.

Algorithm 5.3 Description of reorder_queue () of each ordering metric.

```
{1} breadth first
    do nothing (null operation)
{2} backlink count, $IB`(p)$
    foreach $u$ in url_queue
        backlink_count{$u$] = number of terms $(-,u)$ in links
    sort url queue by backlink count[$u$]
{3} PageRank $IR`(p)$
    solve the following set of equations:
    $IR[u] = (1 -  0.85) + 0.85\Sigma_i(IR[v_i]/c_i)$, where
    $(v_i, u) \in$ links and $c_i$ is the number of links in the page $v_i$
    sort url_queue by $IR(u)$
```

More than 4000 Web pages from different topics like sports, news, education, etc have been downloaded to be used as a data set in the proposed Search Engine. By implementing the basic crawling algorithm the Crawler starts with a seed Web page that is specified by the administrator. Each Web page is assigned a unique identification number (called page_id), then the Crawler begins to find all hyperlinks within the Web page (document) (links ending with .html or .htm).

Each link will be determined and checked by Crawler whether it is retrievable link and its page is able to be indexed or not and to avoid storing the same URL more than one time. Then, this URL will be assigned a unique page_id and stored into the URL_List that functions as a queue. Also, the Crawler extracts important information about the links between two Web pages (source and destination) by parsing the source page. The information which is considered as a link attributes will be valuable in computation of the Page Rank during the Link-Based Ranker phase. These attributes are:

1. **Distance.**

   It is represented between the source Web page and the destination Web page. The distance is determined by if the host-names are different the value of the distance will be equal to 2; the second case is if both host-names are equal the value will be 1.

2. **Visibility of the link**.

   This attribute is determined by checking specific HTML tags which represent the style of the text that is used as a link. These two tags are <B>, and <I>, **bold** and *Italic* style respectively. If the text is either bold or italic the value of link visibility will be equal to 2, otherwise it will be equal to 1.

3. **Position of the link**.

   This attribute is determined by computing the position of URL within the source page. The source page is partitioned into four parts related to the importance of the link because the designer of any Web page would replace the most important link at the beginning of the Web page and so on down to emphasize the importance. The value of link's position will be equal to 4 if the link occurs on the first quarter, 3 in the second quarter lower, 2 in the third quarter, and 1 in the fourth quarter.

Once the Web page has been crawled, files type, and URLs will be stored into the Search Engine Indices. Then the crawler dequeue another Web page from the queue (URL_List) to repeat the process.

The Crawler stops when no more un-crawled Web pages exist. After the Crawler finishes its work, the Indexer and the Link-Based Ranker work simultaneously to save time, using multithreading taking the Web page processed by the crawler directly to the indexer to be processed while the crawler is fetching the next page.

B. Indexer

The indexer is one of the most complicated and critical process in constructing Search Engine. The Indexer is used for analyzing the crawled Web pages content keeping information about each word occurred in an individual Web page. This information is useful to computing the relevancy between the page and the given query. To extract information from the page (document, position, word (d, p, w) triple) to emphasize the importance of the Web page, there exist a number of methods to transform a sequence of bytes, or characters.

Methods that achieve this goal are used together and refer to their collectivity as term extraction techniques. Some of these techniques may omit certain words, change their lexicon, or apply other forms of transformation. Each emitted word ***w*** from the above triple is thus not always an exact match of the word that was encountered in the source text. More pertinently, searching is primarily based on the emitted triples instead of the actual text. It is thus understood that term extraction implies rules that can be used to define formally what a word is in the context of searching a text collection with an inverted index.

Real text is, of course, not just a stream of words. It comes in a wide variety of formats and often includes unwanted special characters. Moreover, natural text contains different words with similar meanings and different forms of the same word, which may be appropriate for consideration when computing a set of answer documents. Translating real text into a stream of words is usually partly dealt with by implementing an appropriate transformation function in the first step of an indexing algorithm. This first step can be organized into the sequence of methods shown below:

- Document Conversion

  It is the first step in translating process which refers to any method that transfers an input text into a sequence of character codes.

- Parsing

  Parsing uses an input sequence of characters to determine word boundaries, and outputs words as (d , p , w). Finding a word boundary can be simple if special characters such as space, full stops, and semicolons are used. These special characters and other such symbols are usually excluded from an index, because their usefulness is limited when searching with methods that are based on the occurrence of words. Determining the end or the beginning of a word can also be a hard task in other situations.

- Word Transformation

  Word transformation concerns different words with equivalent meanings. It is sometimes appropriate to return documents, which contain these equivalent words, as result documents, even if they were not present in the initial query.

In this research, an enhanced Indexer is proposed, which will lead to enhance the quality of the extracted information that must be more accurate to be useful in describing the content of the indexed Web pages. Once the enhanced indexer reads the Web page as a normal text file, it performs the following steps as shown in Figure 5.1:

1. Page Parsing

Most Search Engines are used to convert the whole Web page into a pure text page then parse it partially. For example, Google Search Engine parses only few hundreds of words that lies at the beginning of the page. Any parser which is designed to run on the entire Web must handle a huge array of possible errors. These range from typos in HTML tags to kilobytes of zeros in the middle of a tag, HTML tags are nested hundreds deep, and a great variety of other errors challenge anyone’s imagination to come up with equally creative ones.

In parsing step (in the Enhanced Indexer), each page is fetched from the repository and parsed into a text file which contains the most important HTML tags. These tags are support to the Indexer in the next steps in order to extract the information which is formed, then form accurate attributes for each word occurring in the indexed page.

2. Extracting keywords and their attributes

In this step, after parsing process is completed successfully, the lexical tokens are generated from the text file of the Web pages. Each token is checked whether it is a HTML tag or a real word. If the token is HTML tag it will be processed in the next step, but if the token is a real word a negative dictionary (a dictionary that contains the stop words) is used to remove the noise words that do not reflect the content of the Web page. The negative dictionary (Stop-Word) exists in the Search Engine Indices.

3. Inverted Indexing

In the final step, each prepared token will be processed as in algorithm 5.4: while the reading head does not reach the end of the file the Indexer will read the i_{th} token (where *i* refer to the word's number within the page). If the read token$_i$ is tag it will be processed by the procedure *tag_process*(token$_i$) which extracts the attributes of its related word(s). Each tag has different steps of processing which depends on its type. Table 5.1 represents the processing of important HTML tags.

Algorithm 5.4 Token Processing Algorithm

// Token: set of characters extracted from page.

Input: Tokenized Web page

Output: Inverted Index table

```
{1} while not file.eof do
       if tag($token_i$) then      /* i is the $i_{th}$ number of tokens in the Web page */
          tag_process($token_i$)
        else   if word($token_i$ ) and not stopped($token_i$) then
                   store($token_i$, attribute)
{2}    move to next $token_i$
{3}  end [while]
```

If the token is word and not a noisy word (not *stop*(word$_i$)) then, word descriptor will be extracted, and the word's word_id (identification number of a word in the Lexicon as will be presented in Table 5.5 will be fetched to store the word_id and word descriptor attributes in the Inverted Index Table in Search Engine Indices. In the Inverted Index a mapping between the word_id and the page_id where the word occurs is done.

The information that is extracted in this step can be classified into three classes:

A) Word Descriptor Attributes

These are extracted for each indexed word individually, and stored in the inverted Index Table. These attributes are:

1. Word Position.
2. Word Importance.
3. Word Style.
4. Word Size.
5. Word Font Color.

The format of Inverted Index Table will be illustrated in Table 5.5.

Table 5.1 Tags Processing

Tag Type	**Processing**
<TITLE>	Determine the title of the Web page.
<META>	Determine Meta data set, Author's name, publishing date, modification date, descriptor, and the title if the <TITLE> tag didn't exists in the Web page.
<BODY>	Determine the general text color of the Web page.
<FONT>	Determine the (color, and size) of each word occurred in the Web page.
<H1>...<H6>	Determine the style and the size of each word occurred in the Web page.
<B>, <I>, <U>	Determine the style of the word occurred in the Web page.
Etc…	Etc…

B) Page Information Attributes

More types of attributes are extracted during Indexing and stored at the end of Indexing process. These are stored in Page_Meta_Data Table. These attributes are:

1. Page Title.
2. Author Name.
3. Keywords used in the Web page.
4. Descriptor Terms.
5. Publishing Date.
6. Modification Date.

The format of Page_Meta_Data Table is presented in Table 5.6.

C) Page Descriptor Attributes

Also, another set of attributes is extracted and stored at the end of indexing process. These attributes are related to the form of the Web page and stored in the Page_General_Information Table. These attributes are:

1. Number of words in the Web page.
2. The general font color.
3. The general font size.

The format of this table is presented in Table 5.7.

C. Repository

The repository contains collection of Web pages that downloaded by the Crawler. Each Web page individually is of type (.html or .htm). All these Web pages are indexed by the enhanced indexing system.

D. Search Engine Indices

It is a database that consists of the following tables:

1. Page_Identification Table:

This table is created during the Crawler running interval. The format of this table is represented in table 5.2.

Table 5.2 Page_Identification Table

page_id	page_URL	file_type	crawled
(unique)	**(full address)**	**(.html or .htm)**	**(as 0 or 1)**

This table consists of four fields:

- **page_id**: represents the unique assigned identification number for each individual Web page.
- **page_URL**: represents the full URL of Web page.
- **file_type**: refers to the page's type whether it is .html or .htm.
- **Crawled**: works as an acknowledgement flag to distinguish between the crawled and others which have not yet crawled, because of a modification, or new uploading.

So, this bit is used for reducing the number of revisited pages. Re-crawling or revisiting to the URL list to crawl the pages depends on several *Importance Metrics* that are used to select the starting page. These metrics were described previously. In Crawler a PageRank Importance Metrics is used and the implemented algorithm is represented in algorithms 5.2 and 5.3.

2. **Stop_Word Table:**

Stop_Word Table contains the noisy words. The word in noisy_word field is used to remove the noisy words from the converted page. The format of the Stop_Word Table is presented in table 5.3.

Table 5.3 Stop_Word Table

noisy_ word

noisy_word: represents the set of words which must be removed from the text file, as in table 5.4.

Table 5.4 Noisy_Words

a	and	are	as	at	be	but
for	if	in	into	is	it	by
no	not	of	on	or	as	such
et	that	the	their	then	there	these
they	this	to	was	will	with	etc...

3. Inverted Index Table:

This table is created during the Indexing phase. The information in this table is stored at each word occurrence and this information is used in computing the relevancy of the page depending on every term occurred within the given query (computed by Term-Based Ranker). Table 5.5 presents Inverted Index Table.

Table 5.5 Inverted_Index Table

word_id **(unique)**	page_id **(unique)**	Word	position	importance	style	color	Size

- **word_id:** represents a unique number of each word.
- **page_id:** represents a unique page number that contains the words, whose ids are identified in word_id field.

- **Word:** represents the word extracted from the page.
- **Position:** is the offset of the word within the Web page. The number in this field is related to the length of page (number of words in the page).
- **Importance:** The importance of a word is determined by specifying the location where the word occurs. For example, if the word occurs in the Title of the Web page the word will have a high importance which is6, 4 if it comes in Meta data and 2 other wise.
- **Style:** This attribute represents (Bold, Italic, Underline) styles where each case has its unique identified number. If the style of a specific word is (Bold or Italic or Underline), the word's style value will be equal to 2, else is 1.
- **Color:** represents the font color for the current word. It takes 2 if the color is different than the general color and 1 if it is the same as the general color.
- **Size:** represents the font size for the current word.

4. Page_Meta_Data Table:

The Page_Meta_Data Table contains seven fields as illustrates in table 5.6.

Table 5.6 Page_Meta_Data Table

page_id **(unique)**	page_title	author_name	keywords	descriptor	modification_date	publishing_date

- **page_id:** as described in table 5.5.
- **page-title:** the title is extracted either from the <TITLE> tag or from the <META> tag when its attribute **name** equals title then the content of the **content** attribute represents the Web page title. For example.

  <title> Philadelphia university / Jordan </title>
- **author_name:** the name of the author is extracted from the <META> tag when its attribute **name** equals author then the content of the **content** attribute represents the Web author name. For example,

<META name= “author” content= “Baeza-Yates”>

- **keywords:** the keywords are extracted from the <META> tag when its attribute **name** equals keyword then the content of the **content** attribute represents the most frequently used word which is related to the topic of the Web page.
- **Descriptor:** the descriptor is extracted from the <META> tag when its attribute **name** equals descriptor then the content of the **content** attribute represents a simple description about the subject of the Web page.
- **Modification_date:** Modification date refers to the date that the Web page is uploaded again to the crawler because of any simple modification or updating happening on this Web page. It is extracted from the <META> tag when its attribute **name** equals (update, modfdate, or any word refers to the modification_date), then the content of the **content** attribute represents the Web modification date.
- **Publishing_date:** Publishing date refers to the date that this Web page is published or uploaded at first time to the crawler. It is extracted from the <META> tag when its attribute **name** equals (pubdate, publishdate, or any word refers to the publishing_date), then the content of the **content** attribute represents the Web publishing date.

5. Page_General_Information Table:

This table consists of four fields as illustrated in table 5.7.

Table 5.7 Page_General_Information Table

Page-Id **(unique)**	number of words	color	size

- **Page_id:** as described in table 5.5.
- **Number of words:** refers to the total number of different words occurred in the page_id specified page.
- **Color:** represents the general font color for the current page_id.
- **Size:** represents the general font size for the current page_id.

E. Link-Based Ranker

The ranker system consists of two subsystems, the link-based ranker subsystem and the term-based ranker subsystem. In this section the link-based ranker will be presented. This subsystem is working in off-line time; it depends on the hyperlink structure of the Web page. In this research, several modifications are proposed to achieve a high quality ranking score by implementing the improved PageRank algorithm on the Web pages. These are:

1. Additional Factors Influencing PageRank

The original PageRank calculations are based on the number of inbound links and outbound links only. But by adding the additional factor to the original PageRank equation (4.1) the improved PageRank will achieve more accurate values. Therefore, the following modification of the PageRank algorithm is assumed:

$$PR(A) = (1-d) + d * \left[\left(\frac{PR(T1)}{C(T1)} \right) * IMP(T1, A) + \ldots + \left(\frac{PR(Tn)}{C(Tn)} \right) * IMP(Tn, A) \right] \ldots\ldots(5.3)$$

Where, *IMP*(Ti,A) represents the evaluation of the (inbound or outbound) link which points from page Ti to page A. *IMP*(Ti,A) may consist of several factors, each of which has to be determined only once. Each link has three important factors that are extracted during the crawler running phase. These factors are,

1. Distance between the Web pages which is represented by D in equation (5.4).
2. Visibility of a link which represented by V in equation (5.5).
3. Position of a link within a Web page which is represented by P in equation (5.5).

To implement the evaluation of the linking page into PageRank, the evaluation factor of the modified algorithm must consist of several single factors. For a link that points from page Ti to page A, it can be given as follows:

$$\boldsymbol{IMP\left(Ti,\, A\right) = VP\left(Ti,\, A\right) * D\left(Ti\right)} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(5.4)$$

where D(Ti) represents the distance between Web pages, and VP(Ti,A) is the weight of a single link within a page by its visibility or position, which can be given as follows,

$$\boldsymbol{VP\left(Ti, A\right) = \left(\frac{V\left(Ti, A\right) * P\left(Ti, A\right)}{SUM\left(Ti\right)}\right)} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(5.5)$$

where P(Ti,A) represents the position of a link within a document. P(Ti,A) equals 4 if the link is on the first quarter of the page, 3 if the link is on the second, 2 one third of the page, and 1 if the link is on the forth quarter of the page. V(Ti,A) represents the visibility of a link. V(Ti,A) equals 1 if a link is not particularly emphasized, and 2 if the link is, for instance, bold or italic. SUM(Ti) is the summation of the multiplicative orrelation between V and P, which can be given as follows,

$$SUM\left(Ti\right) = \sum_{K \neq i}\left(P\left(Ti, Tk\right) * V\left(Ti, Tk\right)\right) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(5.6)$$

Additionally, an evaluation of links by the distance between pages as a means to avoid the artificial inflation of PageRank is used because of the bigger the distance between two pages, the less likely has one Webmaster control over both. A criterion for the distance between two pages will be if they are on the same domain or not, distance equals 1 if in the same domain and 2 if they are from different domains. In this way, internal links would be weighted less than external links. In the end, any general measure of the distance between links can be used to determine such a weighting. This comprehends if pages are on the same server or not. This factor is referred to in equation (5.4) by the term D(Ti).

Those factors reflect the probability for the random surfer clicking on a link on a specific Web page. In the original PageRank algorithm, this probability is given by the term (1/C(Ti)), whereby the probability is equal for each link on one page. Assigning different probabilities to each link on a page can, for instance, be realized as follows,

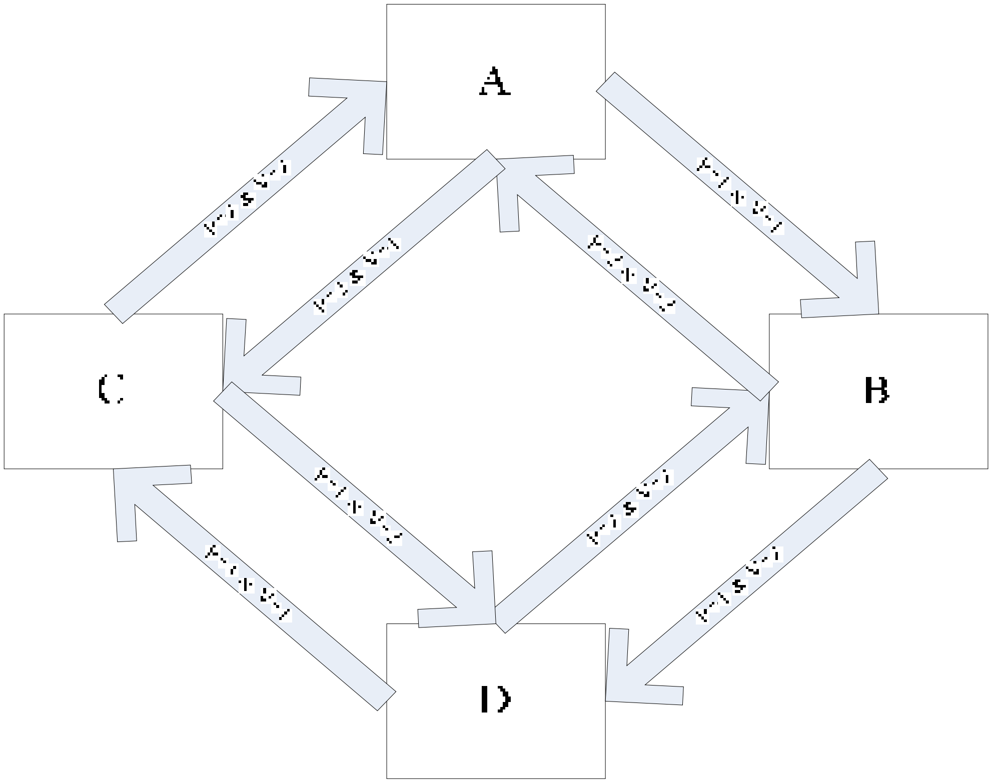


Figure 5.3 Web Pages with its Links Attributes

Consider a Web consisting of four pages A, B C, and D where each of these pages has two outbound links to the near pages as in Figure 5.3. The links are weighted by two evaluation factor V and P. The multiplicative correlation between V and P for the links in this example is evaluated as follows:

V(A,B) * P(A,B) = 1*1 = 1

V(A,C) * P(A,C) = 1*3 = 3

V(B,A) * P(B,A) = 2*2 = 4

V(B,D) * P(B,D) = 1*2 = 2

V(C,A) * P(C,A) = 2*2 = 4

V(C,D) * P(C,D) = 1*2 = 2

V(D,B) * P(D,B) = 2*2 = 4

V(D,C) * P(D,C) = 3*1 = 3

For the purpose of determining the single factors IMP, the evaluated links must not simply be weighted by the number of outbound links on one page, but in fact by the total of evaluated links on the page. Thereby, the following weighting are quotients SUM(T_i) for the single pages T_i:

$$SUM(A) = V(A,B) * P(A,B) + V(A,C) * P(A,C) = 1+3 = 4$$

$$SUM(B) = V(B,A) * P(B,A) + V(B,D) * P(B,D) = 4+2 = 6$$

$$SUM(C) = V(C,A) * P(C,A) + V(C,D) * P(C,D) = 4+2 = 6$$

$$SUM(D) = V(D,B) * P(D,B) + V(D,C) * P(D,C) = 4+3 = 7$$

The result of each *SUM*(T_i) is stored in Page_link_based Table as will be presented in Table 5.9. The evaluation factors VP(T1,T2) for a link which is pointing from page T1 to page T2 are hence given by the equation (5.5). Their values regarding presented example are as follows:

$$VP(A,B) = 1/4 = .25$$

$$VP(A,C) = 3/4 = .75$$

$$VP(B,A) = 4/6 = .67$$

$$VP(B,D) = 2/6 = .33$$

$$VP(C,A) = 4/6 = .67$$

$$VP(C,D) = 2/6 = .33$$

$$VP(D,B) = 4/7 = .57$$

$$VP(D,C) = 3/7 = .43$$

If all pages A, B, and C are in the same site, the evaluation factors *IMP*(T1,T2) for a link which is pointing from page T1 to T2 are given by equation (5.4). Since D(Ti) for all i is equal to (1), the value of *IMP*(T1,T2) will be equal to VP(T1,T2). At a damping factor d of 0.85, the following equations for the calculation of PageRank values will be obtained:

PR(A) = 0.15 + 0.85 * (0.67 * PR(B) + 0.67 * PR(C))

PR(B) = 0.15 + 0.85 * (0.25 * PR(A) + 0.57 * PR(D))

PR(C) = 0.15 + 0.85 * (0.75 * PR(A) + 0.43 * PR(D))

PR(D) = 0.15 + 0.85 * (0.33 * PR(C) + 0.33 * PR(B))

Solving these equations gives the following PageRank values for the presented example:

PR(A) = 1.32

PR(B) = 0.95

PR(C) = 1.39

PR(D) = 0.81

First of all, it is obvious that page C has the highest PageRank of all four pages. This is caused by; page C receiving the relatively higher evaluated link from page A as well as from page D. Furthermore, even by the evaluation of single links, the sum of the PageRank values of all pages almost equals 4 which is the number of pages.

2. Using the modified PageRank value for Web page

The original PageRank equation (4.1) is implemented in several iterations to be as most as converge. So, in the first iteration all pages rank value is equal to ***one*** as an initial value, then in the second iteration pages' rank value at the first iteration is used as an input to the next iteration and so on. Instead of using PR(t)$_{i-1}$ (of i-1$_{th}$ iteration) as input to PageRank equation in the i_{th} iteration, the most recent PageRank can be used as it is clear in this equation:

$$PR(Pi) = (1-d) + d * \left[\sum_{j<i} \frac{PR(Pj)l+1}{C(Pj)} + \sum_{j \geq i} \frac{PR(Pj)l}{C(Pj)} \right] \ldots\ldots\ldots\ldots\ldots\ldots(5.7)$$

By implementing these modifications on the original PageRank equation, the proposed PageRank equation will be as follows:

$$PR(Pi) = (1-d) + d * \left[\sum_{j<i} \left(\frac{PR(Pj)l+1}{C(Pj)} * IMP(Pj,Pi) \right) + \sum_{j \geq i} \left(\frac{PR(Pj)l}{C(Pj)} * IMP(Pj,Pi) \right) \right] (5.8)$$

Where *l* is the iteration number.

The Link-Based Ranker algorithm is shown as in algorithm 5.5.

F. Links

The links in repository consists of two tables:

1. URLs_Information Table:

The contents of this table are created at the crawling phase during the URL extraction operation. The format of this table is presented in Table 5.8.

Table 5.8 **URLs_Information Table**

URL_ID **(unique)**	page_out	page_in	visibility	position	Distance

- **URL_ID:** the serial number of the link
- **page_out**: the page_id of the page that points to another page.
- **page_in:** the page_id of the page that is pointed by the other pages.
- **Visibility:** the style of the object that is used to make the link, (e.g. text or image).
- **Position:** the position of the link within the Web page.
- **Distance:** the distance between the two linked pages. This could be recognized from the host name.

2. Page_Link_Based Table:

The contents of this table are created by the crawler as an initial basic data set to be used by the Link-Based Ranker. The format of this table is present in Table 5.9.

Table 5.9 Page_Link_Based Table

Page_Id **(unique)**	Term_Based_Ranker	SUM(T_i)	Page_Rank

- Term_based_ranker: Shows the number computed by the term based ranker.
- SUM(Ti): it is a numerical factor used by the proposed page_Rank equation (link-based ranking).
- Page_Rank: computed ranking score by link-based ranking.

Algorithm 5.5 Link-Based Ranking Algorithm

```
// P: Web Page.
Input: Webpage, and hyperlink
Output: link-based rank values
no_error = true
While no_error
    For all Pi in Web retrieve_factors(Pi)
      For all Pj points to Pi
$PR(Pi) = (1 - d) + d * \left[ \sum_{j<i} \left( \frac{PR(Pj)l+1}{C(Pj)} * IMP(Pj, Pi) \right) + \sum_{j \geq i} \left( \frac{PR(Pj)l}{C(Pj)} * IMP(Pj, Pi) \right) \right]$
      err = abs[PR(Pi)l - PR(Pi)l + 1]
      If err>0.000001 then
          no_error = false
end[While]
```

Entity relation diagram of the proposed system is shown in Figure 5.4.

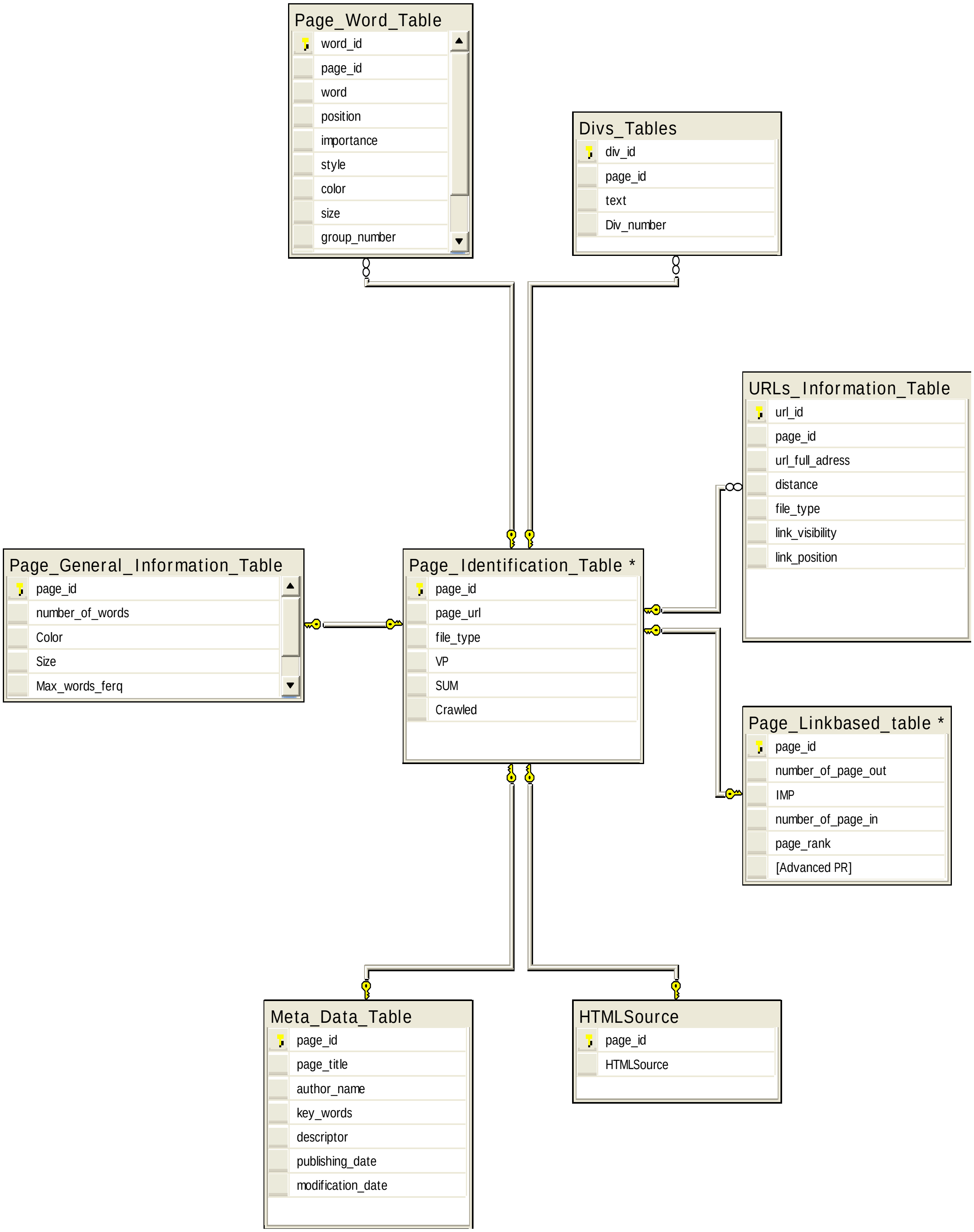


Figure 5.4 System entity relation diagram

5.2.2 On-Line Part

This part depends on the user given a query (it works at Query-Time). It consists of the following subsystems:

A. User Interface

The user interface of the Search Engine consists of two parts: the query interface and the answer interface (HITS). The basic query interface is a box where a sequence of words is entered and has other choices for advanced search.

The proposed Search Engine supports complex query interface, which includes Boolean operators and other features, such as phrase search and title search. For the answer interface, the proposed Search Engine usually returns pages in the order of relevance to the query. In other words, the page that has the higher ranking score appears on the top of the list. Typically, each result entry in the list includes a title of the page, a URL, a brief summary.

The Graphical User Interface (GUI) has no significant changes from any Search Engine design. The GUI is the interface between the user and the back-end of Crawler and Indexer. It is where the search initially begins (initiated by the user) and is also where the search process ends with the search results displayed. The GUI initially, in the normal manner, begins with a search field and a search button. When the user clicks on the search button, the query will be optimized and send to the searcher. Then when the results are ready, they are displayed to the user back on the GUI, with a small brief on the resulted Web Pages.

B. Query Optimizer

This process consists of two steps:

1. Query validation

Once the user posts the search query to the Search Engine, the query will be optimized to provide an error-free query to the Query Ranker. The query is optimized by:

- Removing all additional white characters (Tab, New line, and space). For example, if the posted query was “used   cars   for   sale$”, it will be converted to “used cars for sale$”.
- Removing other special characters like (# , @ , ! , $, ^ , % , …, etc.). For example, the output of the previous step is considered an input to this step so the query which will be “used cars for sale”.
- Eliminating all the noisy word from the query. The final form of the query in the previous example will be “used cars sale”.

2. Query Determiner

The second step in the Query Optimizer is Query determiner that determines the relationship between each two neighbouring terms within the same given query depending on the used operator between them. Even if there is no operator used, in this project the default operator is OR operator. The Stack technique is used in reconstructing the query.

Algorithm 5.6 shows how the query is determined. Once the Query determiner receives the valid query, it will split the query into a set of terms assigned in an array; each cell in the array represents a single term in the valid query. Then, to construct the searched query, each item will be checked whether it is a word or a delimiter. If the item is a word it will be appended to the string of Search_Query. If not, it must be checked if it is a delimiter character [(“) or (”)].

The delimiter character is used to surround the words that are enclosed between them to be matched in the page as a phrase not just as a set of related word. So, if the delimiter character is a beginning delimiter (“) all words in the Op_Stack will be popped sequentially and appended to Search_Query; but if it is an ending delimiter (”) it will be discarded.

C. Searcher

The searcher is the core of relevant pages selection process. It receives the query from the Query Optimizer Subsystem, and then builds SQL statements to search the Inverted_Index Table in the Search Engine Indices for a match with any keywords to retrieve word_id. Depending on the retrieved word_id the following information will be extracted:

1. page_id.
2. position.
3. importance.
4. style.
5. color.
6. size.

Then the Page_General_Information Table is searched for a match with any retrieved page_id to retrieve the following information from the searched table:

7. Number of words (nows).
8. Page font color (color).
9. Page font size (size).

So, each term has its complete information that is stored in a temporary storage buffers. This extracted information is considered as factors used by the Term-Based Ranker in the next step.

Algorithm 5.6 Query Determiner Algorithm

Input: Valid Query

Output: Determined Query

```
{1} receive(Valid_Query)
{2} while not eos.(Valid_Query)  //eos: End of statement (submitted user query).
        $item_i$ =  take_item(Valid_Query)
{3}  end[while]
{4}  for i = 1 to n  /* where n is the number of items within the Valid_Query*/
        if word($item_i$=”intitle”) then
            append(search_query,$item_{i+1}$)
        if word($item_i$=”author”) then
            append(search_query,$item_{i+1}$)
        if word($item_i$) then
            append(search_query, $item_i$)
        else
        if begin_delimiter($item_i$) then
            while not end_delimiter(Op_Stack.top)
                operator = pop(Op_Stack.top)
                append(Search_Query, operator)
            end[while]
 {5}   end[for]
 {6}   end.                    // of the algorithm.
```

D. Term-Based Ranker

Term-Based Ranking is done primarily to provide relevant pages to the user. The non relevant pages will not be retrieved, unless search option includes this information. The Term-Based Ranking algorithm ranks the retrieved Web pages to determine the relevancy of these pages to the query terms. (Equation 5.9).

The proposed Term-Based Ranking algorithm is implemented as shown in algorithm 5.7.

```
Algorithm 5.7            Term-Based Ranking Algorithm
Input: Set of pages
Output: term-based rank values
{1}  for all pages in Retrieved_pages_list
{2}
    $R(i,q) = \sum_{termj \in q} \left( \left( 0.5 + 0.5 \cdot \frac{TF\left(i.j\right)}{TF\ \max\ i} \right) \cdot IDFj\ .. \right)$
{3}    goto {1}
```

$$\boldsymbol{R\left(i,q\right) = \sum \left(\left(0.5 + 0.5 \cdot \frac{TF\ \left(i,j\right)}{TF\ \ max\ \ i} \right) \cdot IDFj \right)} \qquad \text{………} (5.9)$$

where:

N: the number of Web pages in the index database.
$R_{(i,q)}$: the relevance score of the i_{th} page respect to the query q
TF(i,j): the term frequency of Qj in Pi
TF max_i: the maximum term frequency of a keyword in $Page_i$
IDFj:

$$\log\left(N \Big/ \sum_{i=1}^{N} C\left(i,\ j\right)\right)$$

C(i,j) : Documents that contain the query term j in document i.

To determine which Web page is to be displayed first on the result list, the higher number of hits on a Web page determines a higher quality page and will be displayed to the user at the top of the list. This is known as the term-based ranking algorithm. Once the results are gathered the document ranking based on term-based will be multiplied by document ranking based on advanced link-based [R(i,q) * PR(i)] and then pages will be sorted according to the new result in Ranking. The searcher will display them back on the GUI to the user. The URLs of every keyword match are potential results to be displayed to the user.

5.3 System Implementation

On the server side, the process involves creation of a database and its periodic updating done by software called Crawler. The Crawler crawls the WebPages periodically and indexes these crawled WebPages in the database. In the indexing process the stopwords are removed to make the database clear. Then all crawled pages are indexed later. As stated earlier, the Indexer processes the Web pages through the hyperlinks. In this process it extracts the Metadata like ‘title’, ‘keywords’, and any other related information needed for providing complete search results from the HTML document, and the entire content of the HTML document is saved in database.

The idea of indexing the whole page except the stop words is based on the fact that a page dealing with a particular issue will have relevant words throughout its page. Thus indexing all the words in a document increases the probability of getting the relevant WebPages to a query.

The front-end of the Search Engine is the client side having a graphical user interface as in Figure 5.5, which prompts the user to type in the search query. The interface between the client and server side consists of matching the user query with the entries in the database and retrieving the matched WebPages to the user’s machine.

The database consists of a number of tables that are arranged so as to facilitate faster retrieval of the data. This database is housed in a database server that is called Search Engine Indices, which is connected to the Search Engine. The typical English Search Engines will have more than one database server due to the huge number of English Web sites. The proposed Search Engine uses a single database server, because of the small number of Web pages taken from the internet without any interference from the designer contains news, education, sports, entertainment, etc. When the user types the query it is taken to the server housing the Search Engine.

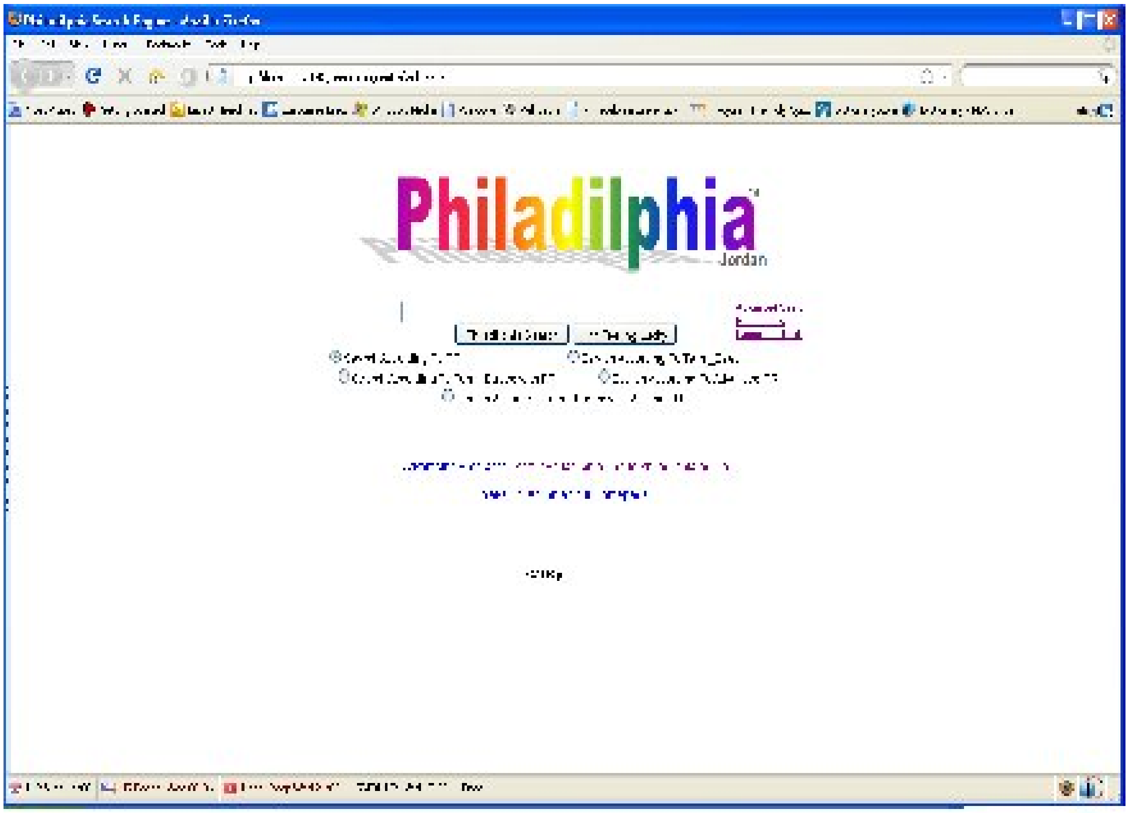


Figure 5.5 Search Engine User Interface

The proposed Search Engine validates the query and then translates it into the structured query language (SQL) which is understandable to the database and passes this SQL query to the Search Engine Indices. The Search Engine Indices identify the database entries that match the query given and sends to the proposed Search Engine server these entries along with other information related to other entries such as the title, the author name, URL and the matching portion from the content of the corresponding entry. The proposed Search Engine sorts these database entries using a ranking algorithm. The ranking algorithm determines the relevancy of a retrieved Webpage to the user query. The retrieved sites are then displayed along with links to these sites and a small portion of text from the matched content. This text gives an idea to the user about the page before the user goes to that particular page.

Advanced Search options allow the user to search for various combinations of the query terms. Some of the search options include Boolean search and phrasal search. By Boolean Search, several options should be available to the user to refine the query. This is important because the search should return only the relevant pages to the user. Boolean search option includes OR, AND, and NOT logic operators, the default being OR operator. Boolean search can be illustrated by the following example. Consider a query consisting of two words. The search results for the OR logic will retrieve the pages containing either of the two terms and the search results for the AND logic retrieve the pages containing both the query terms. The NOT search returns the WebPages that does not contain the NOT term.

The Search Engine automatically searches for both the AND & OR logic. Phrasal search looks for a phrase instead of a word in the database. To include phrase search in the query the user should type the phrase between two quotes (“ ”). The corresponding phrase will be searched as is in the database. This option is particularly useful if the user knows a phrase in the domain of the search. However, this option requires huge processing power and bigger memory in the database as it is represented in Figure 5.6.

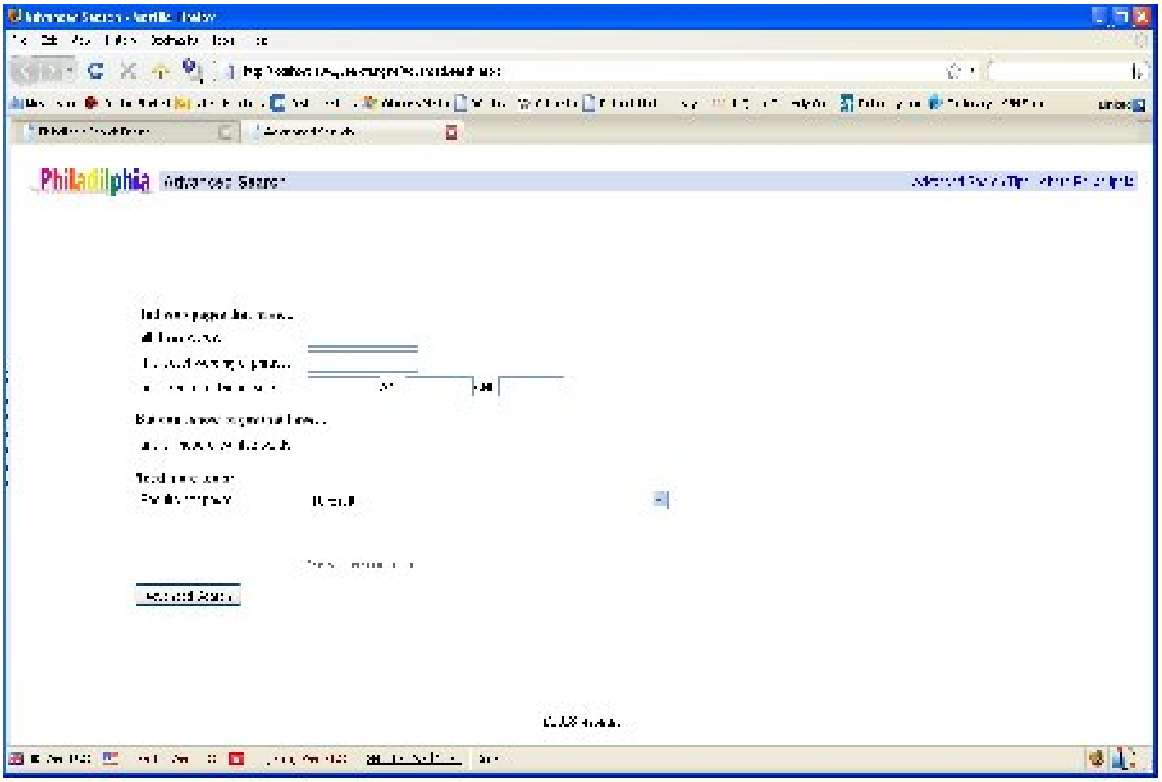


Figure 5.6 Advanced Search Engine User Interface

Chapter Six

Testing and Evaluation

6.1 Introduction

To evaluate any Search Engine, there are two main classes of performance measures, *Effectiveness* and *Efficiency*. Also, from the software viewpoint the *Understandability* of the User Interface is considered an important issue.

6.2 Effectiveness

According to Baeza Yates et al (1999), the determination of how successful the returned search results are is quite a complex process. Two measures that can be used are *Recall* and *Precision*. Which are the most effective ways of measurements beening used by (Text Retrieval Conference) TREC.

Recall is the fraction of the relevant documents (contained in the index) that have been retrieved and precision is the fraction of the retrieved documents that are relevant. However, there are a number of problems with these two measures.

Firstly, they assume that all the documents in the document set have been examined closely. Unfortunately, this is not the case as neither the test candidates nor the author have full knowledge of the document sets' contents.

Secondly, to evaluate these measures accurately the document set should be quite large, this leads to a testing cycle that can be quite time consuming and resources.

To explain these principles (*Recall* and *Precision)*, some examples will be introduced. For a Query *I*, R is the set of all relevant Web pages in existence at a certain point of time. Ra is the set of relevant pages that have been returned. A is the set of all results returned for a search at the same time, as in Figure 6.1.

Recall

Recall determines the percentage of relevant documents that were retrieved. The Recall value is between 0 and 1. It defines as:

$$\textbf{Recall} = \frac{\textbf{\textit{Number of relevant documents retrieved}}}{\textbf{\textit{Number of relevant documents}}} = \frac{\textbf{\textit{Ra}}}{\textbf{\textit{R}}}$$

A high recall means the most of the page that should be returned by a perfect Search Engine is returned. While in normal or in advanced mode all results that are presented in the Search Engine User Interface are still used. The full evaluation of the recall can only be done by doing a user plane review or (by a user judgment).

Precision

Precision is a measure that shows how much of what the user sees is relevant. The resulting value is a real number between 0 and 1. Precision is very important to the proposed Search Engine given thousands of Web pages. This measure is defined as:

$$\textbf{precision} = \frac{\textbf{Number of relevant documents retrieved}}{\textbf{Number of retrieved documents}} = \frac{\boldsymbol{Ra}}{\boldsymbol{A}}$$

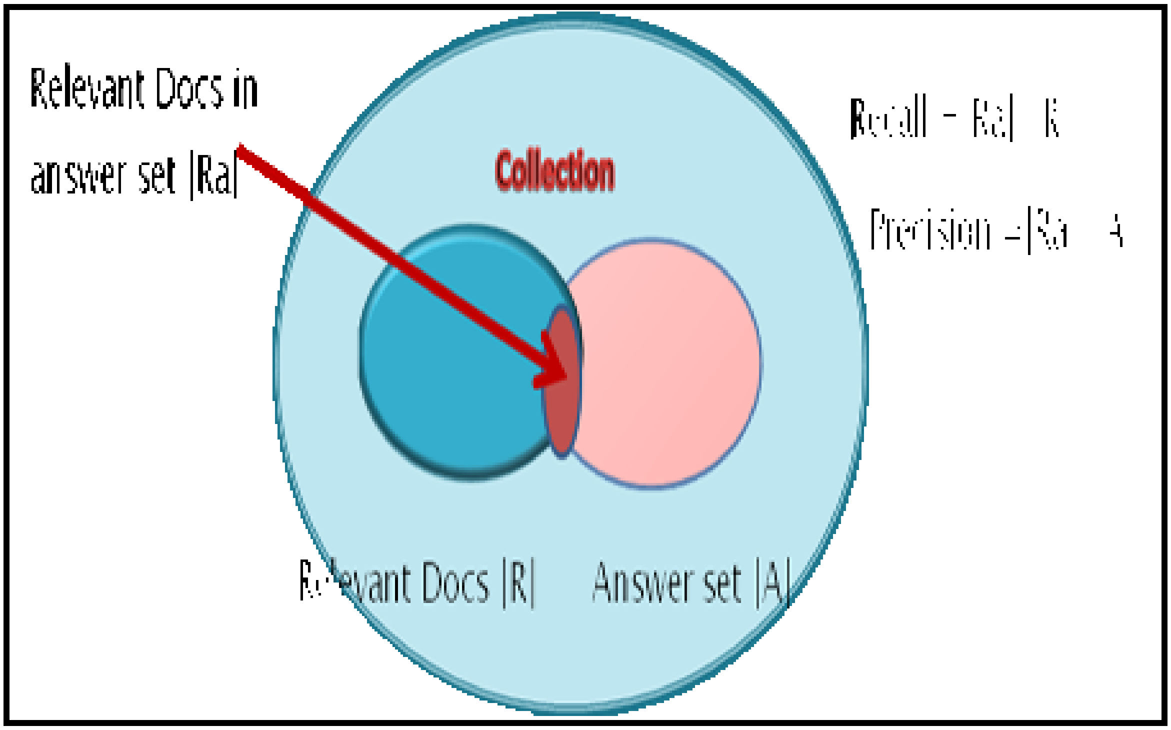


Figure 6.1 Recall And Precision.

In this research, five different kinds of queries are tested and evaluated by Recall and Precision of the first 10 retrieved results. The following results are obtained as illustrated in Table 6.1.

The results show that the general Recall of the first 10 retrieved Web pages is always 100%, which is reasonably good. For the Precision results it is obvious that the rank values of the retrieved Web pages reflect the real relevancy of the existent Web pages in the proposed Search Engine databases.

The proposed Search Engine ranking system effectiveness proved to lead to more relevant document presented at the beginning of the GUI for the user, because as stated before most users examine the first 10 pages, better than the original page ranking system presented by Google by 13%. Figure 6.2 and 6.3 represents the proposed Search Engine results and results obtained using the original page ranking system for the same query respectively.

6.3 Efficiency

Efficiency measures the amount of relevant pages retrieved by the proposed Search Engine in comparison with the original page ranking, term-based ranking, and the time that the system requires to find relevant information. Ten queries have been made on the advanced page ranking system compared by the original page ranking system and term-based ranker, found that the precision of the advanced page ranking is better than the original page ranking and term-based ranking, Figure 6.4, and 6.5 demonstrate the result.

And found that the time needed for the system to produce results for both ranking system almost same with an extra time for advanced ranking system due to the new calculations that should be made to find the results and because of the small database stored in the system. Figure 6.6, 6.7 demonstrates the result.

6.4 Understandability

The proposed system has a user interface that looks like that of any other system as it is illustrated in Figure 5.4. It resembles most of commercial Search Engines interfaces. The only difference is in displaying the results which are the most relevant not related to how much advertisements do the Web site owner pays the Search Engines administration.

The system’s retrieving time is not stable; it depends on the complexity of the given query especially if the query is constructed in the advanced search mode that will take more time.

Table 6.1 Searching result, Recall and Precision measures			
Query	Results	Recall	Precision
Title “al al-bayt university”	All first 4 pages are relevant	100%	100%
phone directory	1,2,3,4,5,6,8,9 Relevant 7, 10 Irrelevant	100%	80%
disease and medicine	All first 7 pages are relevant	100%	100%
Good movie for you	1,2,3,4,7,8,9,10 Relevant 5,6 Irrelevant	100%	80%
cnn news Web site	All 10 pages are relevant	100%	100%

Figure 6.2: The search results in proposed advanced rank

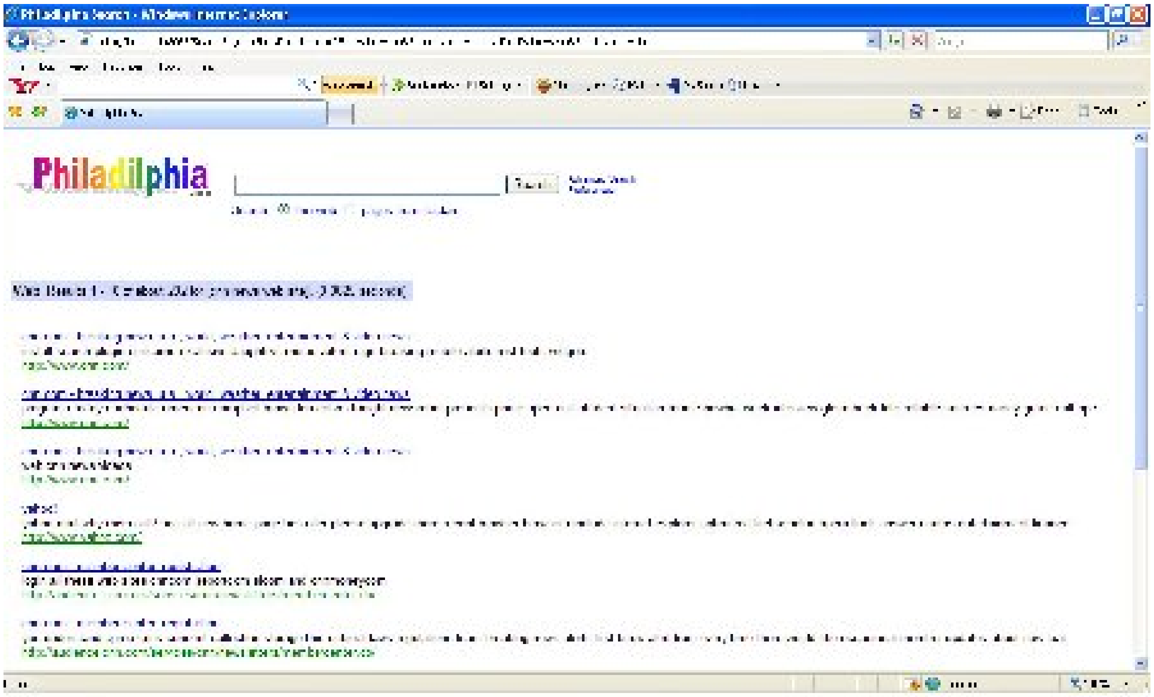


Figure 6.3: The search results (hits) in the original page rank

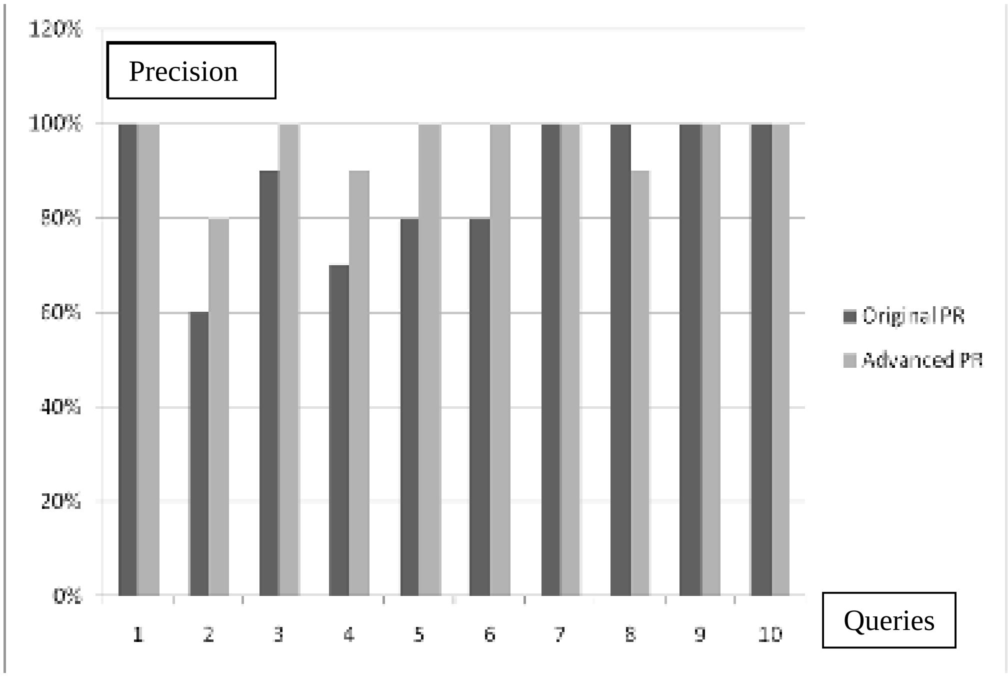


Figure 6.4: Comparison between precision of original PR and advanced PR

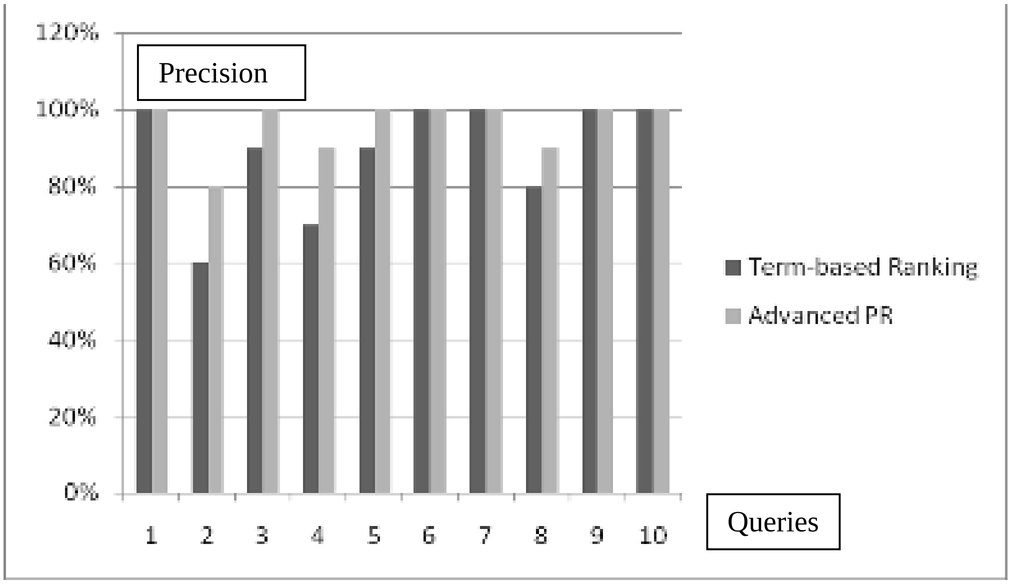


Figure 6.5: Comparison between precision of Term-based PR and advanced PR

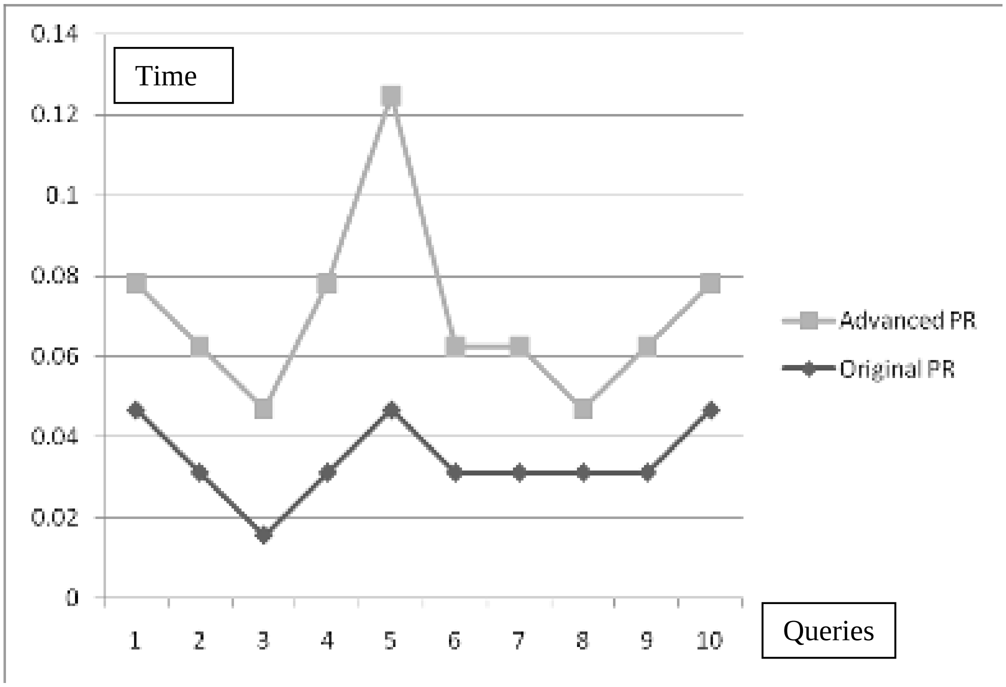


Figure 6.6: Time comparison between the original PR and advanced PR

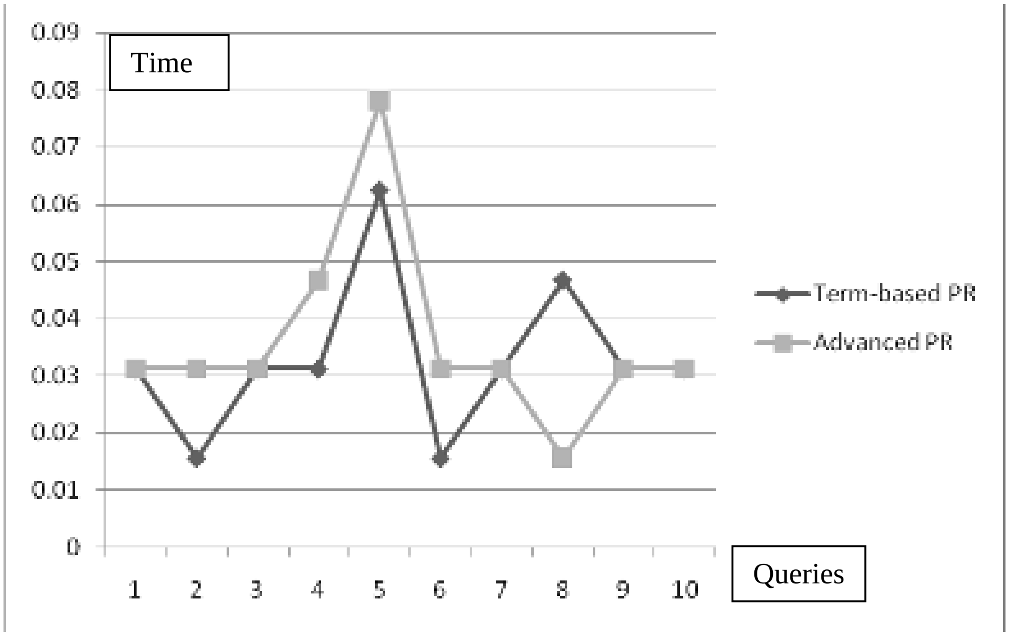


Figure 6.7: Time comparison between term-based PR and advanced PR

Chapter Seven

Conclusions and Future Works

7.1 Introduction

The last chapter discusses the evaluation and the ways that been used to test the enhanced Search Engine through enhancing the indexer and the ranking module. This chapter will conclude the research by reflecting its aims and objectives and detailing any future work that should be considered for Search Engines in general.

7.2 Conclusions

The research aims outlined previously in chapter one were successfully met and achieved. The appropriate literature was reviewed and evaluated; this research addressed the core issues that need to be considered when developing a Search Engine. These issues included indexing, searching, Search Engine interfaces and result set evaluation, given Google Search Engine as a prototype.

It also provides a full understanding for all enhanced indexer criteria's through full indexing of documents. The research proved that the enhanced ranker provide more relevant document than that of the link-based ranker proposed by Brin. S. et al (1998), and a locally hosted Search Engine was designed, developed and tested.

By evaluating the Search Engine results related through different ranking systems as explained in chapter five, the system can use a combination of term-based ranking and link-based ranking giving these conclusions:

1. By feeding the proposed Search Engine with different kinds of queries, the produced results proved that, the aim of this research has been achieved in retrieving the most relevant Web pages to the optimized user query.
2. The index processing takes longer time in indexing the Web page, because in the proposed Search Engine full text indexing processing is performed, not partial indexing.
3. The retrieved documents in this research are limited on Web pages or Web sites only (Files of type *.htm and *.html).

4. The enhanced indexing and enhanced ranking system proved to give the user more relevant documents and improved the precision as stated in chapter 6.

5. Merging the term-based ranker and link-based ranker proved to be a great improvement to the Search Engine recall and precision.

6. By implementing several experiments on Google Search Engine (2008), two important drawbacks were discovered:

   a) Any Search Engine should not care for the stop words in the user query, But Google Search Engine does care. For example, if a given user query is: **"The optimal ranking for Web page"**, Google Search Engine retrieved **372,000** Web pages only. By eliminating or discarding the stop words (the, and for) from the previous user query it became: "**optimal ranking Web page"**, Google Search Engine retrieved **3,350,000** Web pages.

   b) In Google Search Engine, indexing process includes the title and Meta tags information and few hundreds of words that occur in the beginning of the Web page. That means if the user query is a full title of a previously known Web page, Google Search Engine must retrieve it certainly. But by applying several user queries Google Search Engine does not retrieve these Web pages until additional information must be submitted with the user query.

7.3 Future Work

By the experiments, several suggestions are identified that could be implemented in the future to make the project more optimal in its activation with the user:

1. Because of the limitation of this research in retrieving only *.htm and *.html files, in the future, other kinds of textual documents could be retrieved such as *.pdf files.

2. Expanding the ability of the proposed Search Engine in retrieving image files such as *.bmp, and *.jpg, and media files, audio or video such as *.wav.

3. Building a spelling checker to identify the wrong word in the user query, and mistakes corrector to correct or repair the wrong words. This provides a helpful user interface to the user.

4. Different Web services include Search Engine have become a promising technology for E-trading (E-commerce) and for the development of new Internet-based software systems. The new systems should not only look for the required information but also for the quality of the retrieved information and services. Consumers not only expect the service to meet functional aspects but they also demand good quality of services (QoS) such as service reliability, security, trust and execution cost. It is therefore imperative to devise techniques to publish less subjective QoS values to assist service consumers in selecting services according to the desired level of QoS Wei Li Lin et al (2008).

5. Meta-Search Engines are operating on the theory that if using one Search Engine is good, using multiple search programs at one time is even better, it digs out the database of multiple Search Engines and deliver the result as one listing. Meta-Search Engines could provide a unified access for their users, because the required information of users is distributed in the databases of various Search Engines. It is inconvenient and inefficient for an ordinary user to invoke multiple Search Engines and identify useful documents from the returned results. This approach will improve the relevancy of the retrieved documents because there is no Search Engine that can crawl and index the whole Web sites. Amir Hosein Keyhanipour et al (2007), Shengli Wu et al (2004), Engines (1999).

References

AltaVista Search Engine www.altavista.com accessed on 2008.

Amir Hosein Keyhanipour, Behzad Moshiri, Majid Kazemian, Maryam Piroozmand, Caro Lucas (2007), Aggregation of web search engines based on users’ preferences in WebFusion, Knowledge-Based Systems 20, pages 321–328.

Arvind Arasu, Jasmine Novak, Andrew Tomkins, John Tomlin (2002), PageRank Computation and the Structure of the Web: Experiments and Algorithms, <www2002.org/CDROM/poster/173.pdf>.

Baeza-Yates, R., Ribeirio-Neto, B. (1999), Modern Information Retrieval, ACM Press, Addison Wesley.

Braun, Loes (2002), Information Retrieval from Dutch Historical Corpora < http://www.nici.ru.nl/~idak/publications/papers/scripties/scriptiebraun.pdf >.

Brewster Kahle (1997), Archiving the Internet. Extended version of the article Preserving the Internet that appeared in Scientific American, March.

Brian Pinkerton, “Finding What People Want: Experiences with the Web Crawler”. The Second International WWW Conference Chicago, USA, October 17-20, (1994).

Brin, S., Page, L. (1998), The anatomy of a Large-Scale Hypertextual Web Search Engine, Computer Science Department,Stanford University, Stanford, CA 94305, USA.

Budi Yuwono, Savio L. Lam, Jerry H. Ying, Dik L. Lee (1995). A World Wide Web Resource Discovery System. The Fourth International WWW Conference Boston, USA, December 11-14.

BuzzBoltMedia.Com, accessed (2008)< http://www.aboutus.org/Search_Engine >.

C.J. Van Rijsbergen (1979). Information Retrieval. Butterworths,. Available at http://www.dcs.gla.ac.uk/Keith/Preface.html.

Carriere, J. and Kazaman, R. (1997), WebQuery: Searching and visualizing the web through connectivity. In proceedings of the sixth international conference on the World Wide Web, pages 107-117.

Chirita, P.A.; Olmedilla, D.; Nejdl, W. (2003), Finding related hubs and authorities Web Congress, Proceedings. First Latin American Volume, Issue, 10-12 Nov. Page(s): 214 – 215.

Craig Silverstein, Hannes Marais, Monika Henzinger, Michael Moricz (1999), Analysis of a very large web search engine query log, ACM SIGIR Forum, Volume 33 , Issue 1 Pages: 6-12 .

Darnell, R., accessed (2008), HTML Unleashed, Professional Reference http://www.webreference.com/dlab/books/html-pre/43-0.html..

Doyle DJ, Ruskin KJ, Engel TP (1996), The Internet and medicine: Past, present, and future. Yale J Biol Med; 69(5) pages 429–437.

Fuhr, N. (1992), "Probabilistic Models in Information Retrieval". The Computer Journal. 35 (3), pages 243-255.

G. Salton. 1989, Automatic Text Processing. Addison Wesley, Massachusetts.

Google Search Engine www.Google.com accessed on 2008.

Hearst, M.A, (1995). Tilebars: Visualization of term distribution information in full text information access, In Proceedings of the ACM SIGCHI Conference of Human Factors in Computing Systems, (Denver, CO) pages 59-66.

Hsiao, R.L., Technical Reviews accessed (2008), Search Engines Inside Out <http://www.csie.ntu.edu.tw/~b2506023/se.pdf>.

Hsinchun Chen, Chris Schuffels, and Richard Orwig, (1996). Internet Categorization and Search: A Self-Organizing Approach, Journal of visual communication and image representation, Vol. 7, No. 1, March, pages 88–102.

Huang, L., accessed (2008), A Survey On Web Information Retrieval Technologies, CiteSeer: Scientific Literature Digital Library, < http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.12.1902 >.

Jasminka Dobsa, Fakultet organizacije i informatike, Varazdin, Sveuèilište u Zagrebu (2008) , Comparison of information retrieval techniques, < http://videolectures.net/solomon_dobsa_lsici/>.

Junghoo Cho, Hector Garcia-Molina, Lawrence Page (1998). Efficient crawling through URL ordering. Source", Computer Networks and ISDN Systems archive Volume 30 , Issue 1-7.

Junghoo Cho, Hector Garcia-Molina, Lawrence Page (1998). Efficient Crawling Through URL Ordering Computer Networks and ISDN Systems archive Volume 30 , Issue 1-7 Pages: 161 - 172

Karlgen, J. accessed (2008), The basics of information retrieval: Statistical and Linguistics, CiteSeer: Scientific Literature Digital Library, <http://citeseer.nj.nec.com/karlgren00basics.html>.

Kobayashi. M, and Takeda, K. (2000), Information Retrieval on the Web, IBM Research Report, RT0347. ACM Computing Surveys (CSUR) Volume 32, Issue 2 Pages 144 – 173.

Krishna Bharat (2000), SearchPad: explicit capture of search context to support Web search, Computer Networks, Volume 33, Issues 1-6, June 2000, Pages 493-501.

Kyung-Joong Kim, Sung-Bae Cho (2007), Personalized mining of web documents using link, Applied Soft Computing 7 pages 398–410.

Lee Underwood, Search Engine Basics. <http://www.webreference.com/authoring/search_engines/ > accessed (2008).

Liu, B., Zhao, K., and Yi, L. (2002), Visualizing Web Site Comparisons, In proceedings of the eleventh international conference on the world wide web, (ACM Press), (Honanlulu, Hawaii, USA), pages 693-703.

Liu, J., (1999) Guide to Meta-Search Engines <http://www.indiana.edu/~librcsd/search/meta.html>.

Liwen Vaughan (2004), New measurements for search engine evaluation proposed and tested, Information Processing and Management 40, Elsevier, pages 677–691.

M.W. Berry and M. Browne (1999). Understanding Search Engines: Mathematical Modeling and Text Retrieval. SIAM Book Series: Software, Environments, and Tools, June.

Marendy, P., (2001) .A Review of World Wide Web searching techniques, focusing on HITS and related algorithms that utilize the link topology of the World Wide Web to provide the basis for a structured based search technology, CiteSeer: Scientific Literature Digital Library.
< http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.16.8509 >.

MarketingTerms.Com, accessed (2008).<http://www.marketingterms.com /dictionary/search_engine>.

Oliver A. McBryan (1994). GENVL and WWWW: Tools for Taming the Web. First International Conference on the World Wide Web. CERN, Geneva (Switzerland), May 25-26-27.

Piccenelli, G., and Mont, M.C. accessed (2008), A Type 2 Set Based Model for Adaptive Information Retrieval", <http://www.hpl.hp.com/techreports/98/HPL-98-27.pdf>.

Sebrechts, M. M., Vasilakis, J., Miller, M.S., Cugini, J.V., Laskowski, S. J. (1999), Visualization of search results: A Comparative Evaluation of Text, 2D, and 3D interfaces, (SIGIR '99), (CA, USA, ACM), pages 3-10.

Shengli Wu, and Jieyu Li, (2004). Effectiveness Evaluation and Comparison of Web Search Engines and Meta-search Engines, Springer Berlin / Heidelberg, Volume (3129/2004) pages 303-314.

Soboroff, I. accessed (2008), Information Retrieval, <http://www.csee.umbc.edu/~ian/irF02/lectures/03Text-Processing.pdf>.

The Web Robots Page accessed (2008), <http://www.robotstxt.org/wc/robots.html>.

Wei-Li Lin, Chi-Chun Lo, Kuo-Ming Chao, Muhammad Younas (2008). Consumer-centric QoS-aware selection of web services, Elsevier Journal of Computer and System Sciences 74 pages 211–231.

Wilkinson, R., Hingston, P. (1991), Using the cosine measure in a neural network for document retrieval, In proceedings of the ACM SIGIR Conference on Research and Development in information Retrieval, (Chicago, USA, ACM) , pages 202-210.

Witten , I.H., Moffat, A., Bell, T.C. (1999), Managing Gigabytes – Compressing and Indexing Documents and Images, Morgan Kauffman ACM ISBN 1558605703.

Yahoo Search Engine www.yahoo.com accessed on 2008.