A

Dissertation

On

Context Detection in Web Queries.

Submitted in Partial fulfilment of the requirements
For the award of Degree of

MASTER OF ENGINEERING
(Computer Technology and Application)
Delhi University, Delhi

Submitted By:
SANDEEP RAI
(University Roll No.12212)

Under the Guidance of:
Mrs. AKSHI KUMAR
Department Of Computer Engineering
Delhi College of Engineering, Delhi

DEPARTMENT OF COMPUTER ENGINEERING
DELHI COLLEGE OF ENGINEERING
DELHI UNIVERSITY
(2008-2009)

CERTIFICATE

DELHI COLLEGE OF ENGINEERING
(Govt. of National Capital Territory of Delhi)
BAWANA ROAD, DELHI - 110042

Date: _____

It is certified that the work contained in this dissertation titled ***"CONTEXT DETECTION IN WEB QUERIES"*** by ***SANDEEP RAI*** is submitted in the partial fulfilment for the requirements of the degree of ***Master of Engineering in Computer Technology & Application*** at Delhi College of Engineering. This work was completed under my supervision and guidance during the academic session 2008-2009. He has completed the work with utmost sincerity and diligence.

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

Mrs. AKSHI KUMAR
Lecturer
Department of Computer Engineering
Delhi College of Engineering, Delhi-42

ACKNOWLEDGEMENT

I am thankful to my Almighty for the blessings that were bestowed on me to make this work possible. It is a great pleasure to have the opportunity to extent my heartfelt gratitude to everybody who helped me throughout the course of this project.

It is distinct pleasure to express my deep sense of gratitude and indebtedness to my learned supervisor Mrs.Akshi Kumar for her invaluable guidance, encouragement and patient reviews. Her continuous inspiration has made me complete this dissertation. She kept on boosting me time and again for putting an extra ounce of effort to realize her work.

I would also like to take this opportunity to present my sincere regards to my teachers Dr. Daya Gupta, Mrs. Rajni Jindal, Mr. Manoj Kumar, Mr. Vinod Kumar and Dr. S.K.Saxena for their support and encouragement. I am thankful to the Computer Centre Head, Mr. Manoj Sethi and other staff members for providing me unconditional and anytime access to the resources.

I am grateful to my parents, brother and sister for their continual support and encouragement prior to the commencement of this work, during and now after.

Above all, I thank my classmates for their unconditional support and motivation during this work.

SANDEEP RAI

M.E. (Computer Technology and Application)

Department of Computer Engineering

Delhi College of Engineering, Delhi-42

ABSTRACT

On the Web, the most commonly used tool for learning is the search engine [7]. The user first submits a query representing the 'subject of interest' to a search engine system, which finds and returns the related Web pages. He/she then browses through the returned results to find those suitable Web pages. Search engines are critically important to help users find relevant information on the World Wide Web. In order to best serve the needs of users, a search engine must find and filter the most relevant information matching a user's query, and then present that information in a manner that makes the information most readily palatable to the user. However, the current search techniques are not designed for in-depth learning on the Web.

Contextual search refers to proactively capturing the information need of a user by automatically augmenting the user query with information extracted from the search context; for example, by using terms from the web page the user is currently browsing or a file the user is currently editing. We implement a novel context detection algorithm given a user's query. The aim is to use "Context as a query" and treat the context as a background for topic specific search [1]. Thus we try to find a possible solution set for the following questions:

1) How to detect the context contained in the user query?

2) How to possibly refine the query at the contextual level?

Table of Content

List of Tables

List of Figures

Chapter 1

Introduction

In this chapter we introduce the Web Information Retrieval paradigm. We expound the impact of Web and the opportunities, challenges that make it an active area of research. The motivation behind the work done & the scope is described.

1.1 **Motivation**

With the explosive growth of the World Wide Web, we are currently facing new circumstances in the Web data oceans. Current Web searching engines, mainly based on traditional IR technologies, are insufficient to truly meet users' information needs. We foresee that the biggest challenge in the next several decades is how to effectively and efficiently dig out the knowledge from huge amounts of the Web data. Web Information Retrieval is defined as the application of theories and methodologies from IR to the World Wide Web. It is concerned with addressing the technological challenges facing Information Retrieval (IR). The characteristics of Web make the task of retrieving information from it quite different from the Pre- Web (traditional) information retrieval. As a result, there has been a rapid growth in the area of Web information retrieval research, which focuses on automatically discovering information and knowledge through the analysis of Web contents, Web structure and Web usages. Since the Web is huge, heterogeneous and dynamic, Web information retrieval calls for novel technologies and tools, which may take advantage of the state-of-the-art technologies from various areas, including machine learning, data mining, information retrieval, database and nature language processing.

A recent Forrester Research report showed that 80% of Web surfers discover the new sites that they visit through search engines. (Such as Ask, Google, MSN or Yahoo). Therefore, search engines have been established as revolutionary working metaphors.

Web search engines generally treat search requests in isolation. The results for a given query are identical, independent of the user, or the context in which the user made the request. Next-generation search engines make increasing use of context information, either by using explicit or implicit context information from users, or by implementing additional functionality within restricted contexts. Greater use of context in web search helps in increasing the competition and diversity on the web. Context-based retrieval approaches aim to provide a more complete retrieval process by incorporating contextual information into the retrieval process. The use of context in information retrieval is not a new idea. Unfortunately none of them proves to become a Silver Bullet, at least so far. Moreover, these approaches are often combined to achieve better performance and recall/precision of information retrieval.

1.2 Problem statement

Our aim is to proactively capture the information need of a user by automatically augmenting the user query using contextual information. It is further defined with the help of the following research goals that are identified for this purpose:

1) How to detect the context contained in the user query?
2) How to possibly refine the query at the contextual level?

1.3 Scope

The scope of this research is circumscribed to context detection in web queries system based on text by the snippet.

1.4 Organization of the Remainder of Thesis

This thesis is organized into 5 chapters followed by references and appendices.

Chapter 2 gives a review of the relevant & related Web Information Retrieval literature most related to the aim of this research. It gives an overview of the Web Information Retrieval (WebIR) paradigm, its components and reviews a variety of techniques, approaches & issues related. It gives a brief introduction to the use of

Information Theory in WebIR. The advent of the field of Contextual Retrieval is sketched out.

Chapter 3 presents the framework used and the details of algorithm used. The basics of experimental setup and the graphical user interface representation are discussed in Chapter 4. Chapter 5 forms the conclusion of this thesis and outlines the direction of future research.

A section listing the references used in this thesis follows chapter 5. Appendix A depicting the screen-shots of the system follows.

Chapter 2

Literature Survey

This chapter details out the related and relevant literature. We explain the Anatomy of WebIR, its tasks, tools, components & performance measures. The Basic of Information Theory pertinent to WebIR is also reviewed. Finally the era of Contextual Retrieval is reviewed.

2.1 Anatomy of Web Information Retrieval (WebIR)

Retrieving information from the Web is becoming a common practice for internet users. However, the size and heterogeneity of the Web challenge the effectiveness of classical information retrieval techniques. For the information retrieval (IR) community, the Web now presents a new paradigm, while also generating new challenges and attracting growing interest from around the world. Web IR can be defined as the application of theories and methodologies from IR to the World Wide Web. It is concerned with addressing the technological challenges facing Information Retrieval (IR) in the setting of WWW [3].

(a) Traditional web IR

In traditional IR documents have been represented in the so called vector space model. Documents are tokenized in words, some terms are possibly filtered against a static defined stop–list, and sometimes they are stemmed to extract a canonical form, and represented as a vector in Euclidean space. Each canonical token represents an axis in this space, and each document is a vector in the space. If the term t appears n(t, d) times in document d, then the t–th coordinate of d is just n(t, d).

Traditional techniques, involve such as Query Expansion [29] and Statistical Modeling [43], as well as examining the structure and meta-data of the documents, or analyzing the hyperlinks between the documents.

(b) Modern Web IR

Modern Web IR is a discipline which has exploited some of the classical results of Information Retrieval developing innovative models of information access. A recent Forrester Research report showed that 80% of Web surfers discover the new sites that they visit through search engines. (Such as Ask, Google, MSN or Yahoo). Therefore, search engines have established as a revolutionary working metaphor. If someone needs information about a book, an address, a research paper, a flight ticket, or almost any other topic, they just make a query on a search engine the interested reader can refer to [10, 11, 12]. In this paragraph we briefly review the architecture of a typical search engine.

The goal of a modern WebIR is to retrieve documents considered "relevant" to a user query from a given collection. Nowadays, a user query is modeled as a set of keywords extracted from a large dictionary of words; a document is typically a Web page, pdf, postscript, doc file, or whatever file that can be parsed into a set of tokens. Global search engines serve as de facto Internet portals, local search engines are embedded in numerous individual Web sites, and browsing is the most common activity on the Web, due to the hyper-linked structure that provides access to a large quantity of information in a restricted space.

Thus, WebIR is different from classical IR for two kinds of reasons: concepts and technologies [8]. The following characteristics of the Web shape up the nature of Web Information Retrieval and are what make it considerably different to traditional retrieval challenges [1]:

- ***The "Abundance" of Web*** With the phenomenal growth of the Web, there is an ever increasing volume of data and information published in numerous Web pages. According to worldwidewebsize.com, the indexed Web contains at least 21.82 billion pages (Sunday, 28 June, 2009)
- ***Heterogeneity***

  • Information /data of almost all types exist on the Web, e.g., structured tables, texts, multimedia data, etc.

  • Much of the Web information is semi-structured due to the nested structure of HTML code.

  • Much of the Web information is linked

• Much of the Web information is redundant

• The Web is noisy: A Web page typically contains a mixture of many kinds of information, e.g., main contents, advertisement, navigational panels, copyright notices.

- ***Dynamics*** The freedom for anyone to publish information on the web at anytime and anywhere implies that information on the Web is constantly changing. It is a dynamic information environment whereas traditional systems are typically based on static document collection.
- ***Duplication*** Several studies indicate that nearly 30% of the web's content is duplicated, mainly due to mirroring.
- ***Users Search Behavior*** The users have different expectations and goals such as Informative, Transactional and Navigational. Often they compose short, ill-defined queries and impatiently look for the results mainly in the top 10 results.

2.1.1 Tasks

Web Information Retrieval research is typically organized in tasks with specific goals to be achieved. Existing tasks have changed frequently over the years due to the emergence of new fields. Below is a summary of the main tasks and also of the new or emerging ones [1].

- **Ad-Hoc Retrieval** Rank documents using non-constrained queries in a fixed collection. This is the standard retrieval task in Web IR.
- **Filtering** Select documents using a fixed query in a dynamic collection. For example, “Retrieve all documents related to ‘Research in India’ from a continuous feed”.
- **Topic Distillation** Find short lists of good entry points to a broad topic. For example, “Find relevant pages on the topic of Indian History”.
- **Homepage Finding** Find the URL of a named entity. For example, “Find the URL of the Indian High Commission homepage”
- **Adversarial Web IR** Develop methods to identify and address the problem of web spam, namely link spamming that affect the ranking of results.
- **Summarization** Produce a relevant summary of a single or multiple documents.

2.1.2 Tools

Automated methods for retrieving information on the Web can be broadly classed as search tools or search services.

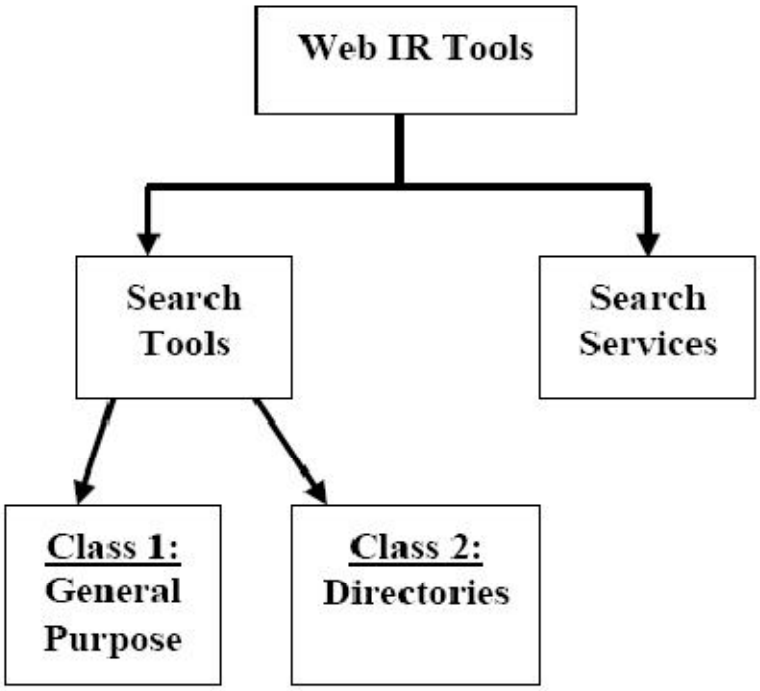


Figure1. Classification of Web IR tools [1]

- **Search Tools**

A Web search engine is a tool designed to search for information on the World Wide Web. The search results are usually presented in a list and are commonly called hits. The information may consist of web pages, images, information and other types of files. Some search engines also mine data available in databases or open directories. Unlike Web directories, which are maintained by human editors, search engines operate algorithmically or are a mixture of algorithmic and human input.

The Search tools employ robots for indexing Web documents. They feature a user interface for specifying queries and browsing the results. At the heart of a search tool is the search engine, which is responsible for searching the index to retrieve documents relevant to a user query. Search tools can be distinguished into two categories on the transparency of the index to the user. The two class categories are depicted along the following dimensions:

- Methods for Web navigation,
- Indexing techniques,
- Query language or specification scheme
- for expressing user queries,
- Strategies for query-document matching, and
- Methods for presenting the query output.

Class1 search tools: General Purpose Search Engine

These tools completely hide the organization and content of the index from the user. Example: AltaVista, Excite, Google, Info seek, Lycos

Class 2 search tools: Subject Directories

These feature a hierarchically organized subject catalog or directory of the Web, which is visible to users as they browse and search. Example: Yahoo!, WWW Virtual Library and Galaxy.

- **Search Services**

The Search services provide users a layer of abstraction over several search tools and databases and aim at simplifying the Web search. Search services broadcast user queries to several search engines and various other information sources simultaneously. Then they merge the results submitted by these sources, check for duplicates, and present them to the user as an HTML page with clickable URLs. Example: MetaCrawler [1].

2.1.3 Components and Their Problems

The search query and analyze some of the problems arising when doing data processing, data indexing and query processing. Based on the data-flow in a standard search engine, some common elements are needed to construct a search engine. The working is as follows:

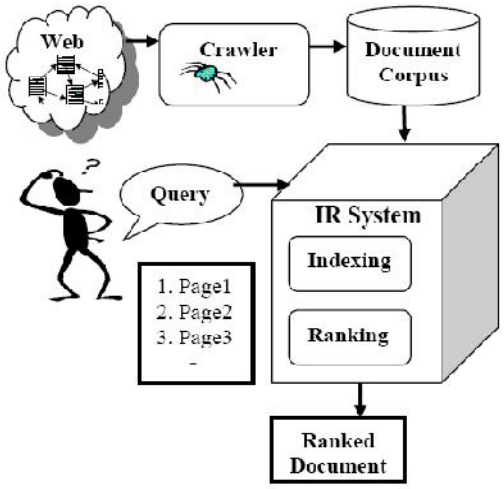


Figure 2. Overview of components of a typical Web Information Retrieval System [1]

1. First a *WebCrawler* is used to download pages from the web and store them in the search engine's database. These pages are defined as documents.
2. When data is present in the database, a *Document Processor* parses the documents and formats them before indexing can take place.
3. An *Indexing* Service takes the parsed and formatted data and creates an index. The Indexing Service only indexes items that have been identified as relevant4 thus make the data ready to be searched. These items are defined as terms.
4. Finally, a *Query Processor* processes the queries from users and searches the database for matches and presents the relevant results to the user.

Based on the above, the construction of a search engine can be split into three different categories:

• Document Processing
• Data Indexing
• Query Processing

The Document Processing deals with data preprocessing, Data Indexing handles the appropriate indexing of the downloaded data and Query Processing submits queries and retrieves the results from the search engine. These elements will be discussed in more detail in the following sections.

(a) Document Processing

Since text cannot be directly interpreted by a search algorithm, an indexing procedure needs to map the text to a proper representation of its content. The document processor does exactly that. It prepares, processes, and inputs the documents (web pages or sites) that the search engine will add to its index. Many problems arise when processing large amounts of data. Even the simplest tasks can become complicated when data is of enormous size. Further, the parsing of data is difficult as there is very little (if any) structure in the various documents on the Internet. Just about anything goes “out there”.

When a document processor is implemented, various problems arise regarding parsing and identifying index able elements (terms). This section discusses how these problems could be addressed.

How a document processor represents text is a choice of which elements of the text it finds meaningful (lexical semantics) and what combinational rules it finds meaningful for these elements (compositional semantics). Usually, the compositional semantics of text is disregarded [18] and text is represented as a histogram of terms that occur in the text, hence only keeping the lexical semantics. To create the term histogram, the document processor performs some or all of the following steps

1. Normalizes the document stream to a predefined format.
2. Breaks the document stream into desired retrievable units.
3. Identifies potential indexable elements in documents.
4. Deletes stop words.
5. Reduces terms to their stems.
6. Extracts index entries.
7. Computes term weights.

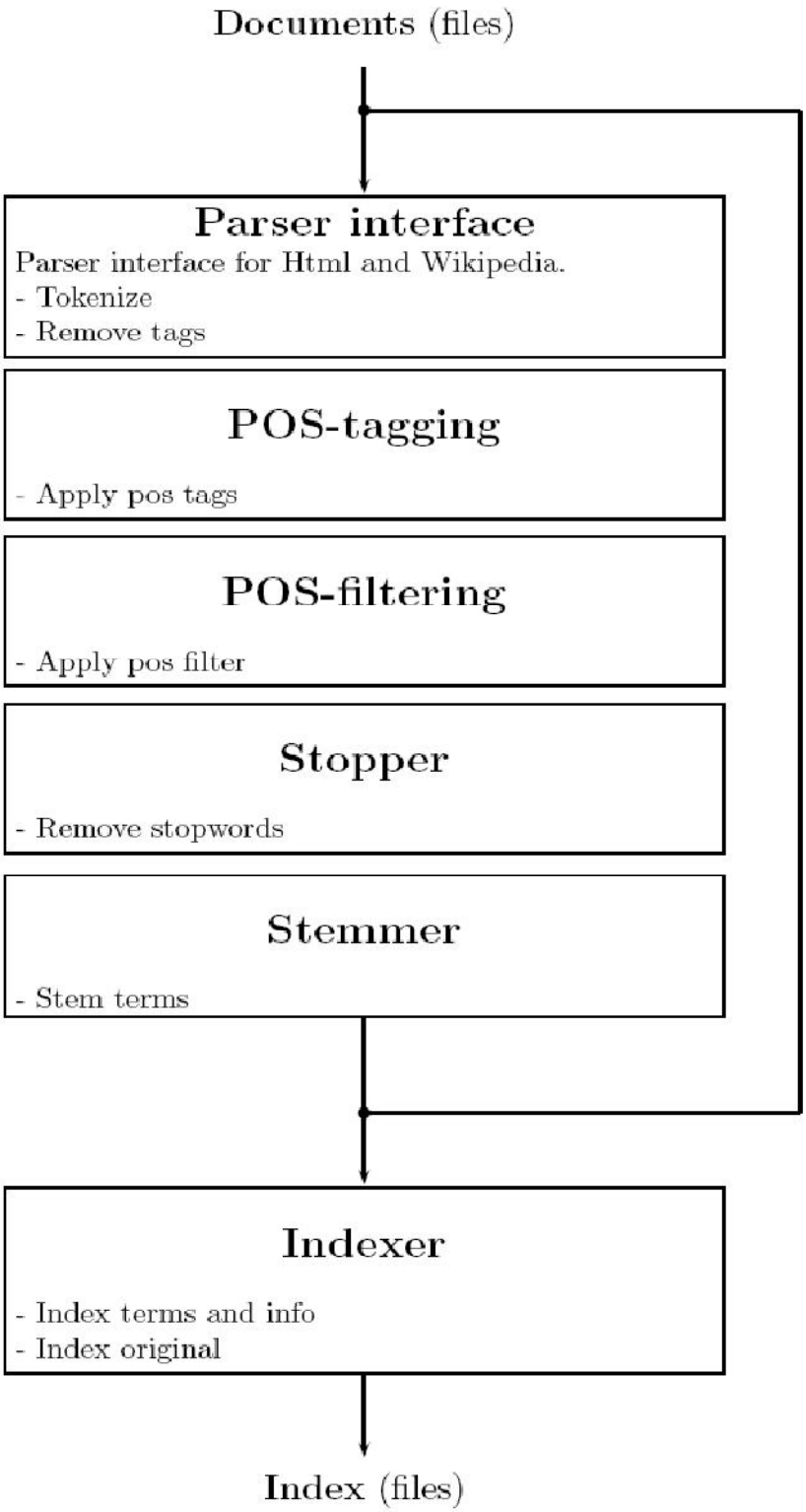


Figure 3. Document processing

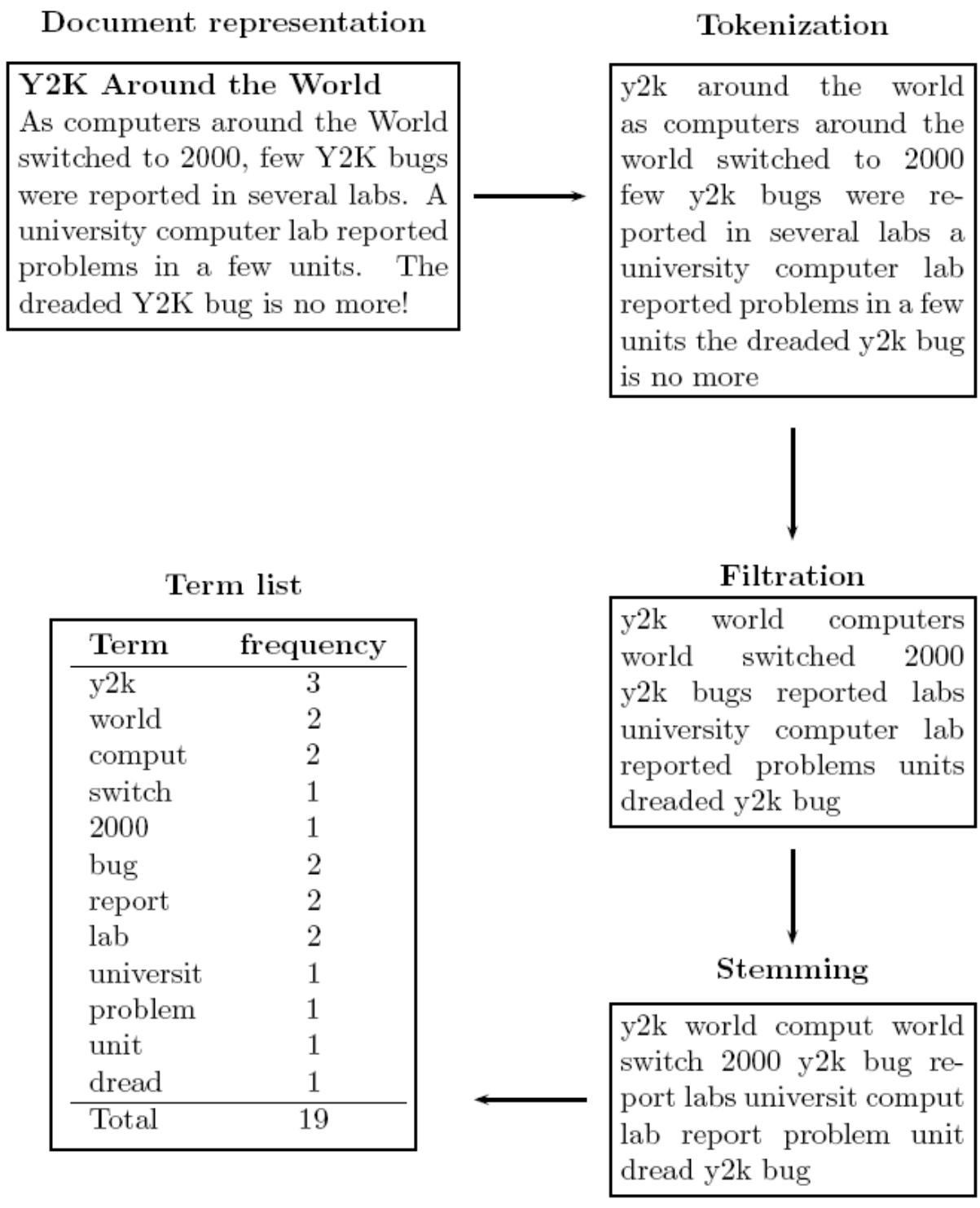


Figure 4. Important steps in the document processor

These steps are very important in the process of creating a search engine as the terms the document processor identifies are the terms that later can be v searched for. V Text is extremely dynamic and can contain hundreds of thousands of characters, symbols, punctuations, digits etc. As a result there are many problems that need to be addressed when dealing with such data and trying to identify indexable elements. Besides the different parsing rules and handling of data, the document processor needs to ease the load of later calculations by only selecting the terms that are important and relevant without losing too much valuable information. This trade off is one of the real challenges in implementing a good document processor.

- **Normalizing and Identifying Indexable Terms**

  First of all the text has to broken into terms and the obvious way would be to break on white spaces - i.e. define a single word as a term. This approach is called bag-of-words as text is seen as a bag of words, thus disregarding compositional semantics. Punctuations can divide sentences but are also used in many other contexts, e.g. variable names in program code such as ’x.id’ or in ’510B.C.’Punctuation cannot be removed uncritically from a sentence thus giving ’xid’ and ’510BC’. One solution could be to create punctuation groups such that punctuations would be replaced by special character sequences, e.g. ’<x>’ in the index giving ’x<x>id’ and ’510B<x>C’ making searches on ’x:id’, ’x;ix’, ’x.id’ etc. mean the same thing. The groups could be refined even further by generating several groups as shown in table 2.1, thus minimizing the number of indexed terms.

Punctuation example	Result
x:id, x;ix, x.id	x<a>id
x’id, x”id, xîd	x<b>id
x(id, x[id, x{id	x<c>id
x((id, x[(id, x{(id	x<c>id

Table1. Punctuation groups

To avoid losing too much of the individual words and adding to processing complexity it is advised to use white space as delimiters and remove punctuations only if a word begins or ends with a punctuation.

When finding indexable terms it is difficult to say when a word or group of words is important enough to be indexed. The case of the word says a lot about its importance and a decision whether ’PET’ and ’pet’ should mean the same thing has to be made. An easy approach is to add more importance to upper case words as they tend to be abbreviations of organizations, terms and/or concepts.

Nouns usually carry most semantic weight and other word groups could be removed. In [19] it is suggested that nouns situated side by side could be placed in noun groups (or compound term) and not just single terms. For example a sentence like ’In computer science we usually...’, ‘computer science’ could be indexed as a single compound term. Syntactic distance between nouns, meaning the distance between two nouns where they are still considered a compound term, is suggested to be at a threshold of three. Only relying on nouns seems to disregard too much information and a term histogram of a mix of terms and compound terms is therefore preferred. Digits and dates present yet another problem. In some cases it might be very useful to be able to search for a given year or date. Some dates and years are very important - e.g.: ’9/11’. Dates and years could be normalized and indexed, although it would require some parsing as dates can be in many formats. If digits are removed, searching for a specific year is not possible, but most of the time that does not make a lot of sense anyway as a year normally cannot be connected with a single searchable item e.g. search query ’2000’ makes sense. One idea is to treat years as nouns and thus include them in the compound terms. If a noun and a year are syntactically close, the year may be important - e.g. searching for ’exploration 1492’. The danger of including digits is that it could lead to an extremely long term list in the index if documents contain a lot of distinct numbers. Spell checking all terms and compound terms found is very difficult. Even if it is assumed that names start with a capital letter so they can be identified, there are plenty of other character constellations that do not fit spelling rules. Again, the program variable ’x.id’ is a

good example. No dictionary would allow such a term. Misspellings might be removed by their frequency, i.e. that terms which appear less than a given minimum threshold are removed (hopefully misspellings are rare). Likewise, very common terms could be removed if they appear more than a given maximum threshold.

- **Stemming**

Stemming is used to remove word suffixes (and possibly prefixes). This reduces the number of unique terms and gives a user's search query a better recall. If taking the classical example from textbooks on stemming, words such as analysis, analyzing, analyzer and analyzed all stem to 'analy'. This example shows that stemming introduces an artificial increased polysemy. Without stemming the term histogram could grow to an unmanageable size. [20] Compares benefits from eight stemming projects and finds conflicting results. Therefore, some search engines do not use stemming at all. [21] Reports that stemming on average increases performance by 1-3% compared to no stemming and for some queries even better. Further, it is shown overall that prefix removal reduces the result yet specific queries perform better with prefix removal. When using stemming, it is not advised to find a word's 'true' root. Linguistic stemmers are simply not good enough to make such a stemming efficiently at this time.

.

- **Term Weighting**

The idea of term weighting is to give terms different weights when situated in different contexts. For example, if a term is located in a title tag, the term is assigned more weight than a term in a paragraph. Another way is simply to use the term frequency (how many times does the term appear in a document) or a combination of the two. There exist many term weighting algorithms, but the one generally used and with consistently good results is the term frequency-inverse document frequency (Tf × Idf). Tf × Idf makes three basic assumptions [22].

1. Rare terms are no less important than frequent ones

2. Multiple appearances of a term in a document is no less important than single appearances

3. Long documents are no more important than short ones

Together these assumptions constitute normalized Tf ×Idf. Term frequency is simply the number of times a given term appears in a document divided by the number of terms occurring, and the inverse document frequency is a measure of the general importance of a term.

- **Web Crawler**

A Web crawler is a computer program that browses the World Wide Web in a methodical, automated manner. Other terms for Web crawlers are ants, automatic indexers, bots, and worms[1] or Web spider, Web robot, or—especially in the FOAF community—Web scutter[2].

This process is called Web crawling or spidering. Many sites, in particular search engines, use spidering as a means of providing up-to-date data. Web crawlers are mainly used to create a copy of all the visited pages for later processing by a search engine that will index the downloaded pages to provide fast searches. Crawlers can also be used for automating maintenance tasks on a Web site, such as checking links or validating HTML code. Also, crawlers can be used to gather specific types of information from Web pages, such as harvesting e-mail addresses (usually for spam).

A Web crawler is one type of bot, or software agent. In general, it starts with a list of URLs to visit, called the seeds. As the crawler visits these URLs, it identifies all the hyperlinks in the page and adds them to the list of URLs to visit, called the crawl frontier. URLs from the frontier are recursively visited according to a set of policies.

(b) Data Indexing

As search engines' indexes have grown very fast in the past few years, the Data Indexing part is becoming the most important part of the data processing within a search engine. If data is not indexed properly, the search engine cannot be expected to yield good results to search queries. In the previous section we discussed what steps need to be considered in the data preprocessing, prior to the actual indexing and as a result the document processing plays a very important role in the data indexing. The results from the document processing should have identified which terms are to be indexed, but the data

indexer must take the final decision of what should be included and what data can be excluded. The naive approach of simply indexing every term that occurs within the downloaded documents would require enormous amounts of data storage, and could also result in a slow response time to queries due to computational inefficiency. In this section we discuss how terms can be indexed in order to get positive results to search queries.

- **Traditional Indexing**

A popular way of creating an index for search engines is to add the terms from the document processor to the index, sometimes calculate term-proximity within the documents and add that information to the index. This way, it is possible to search for combinations in the documents, e.g. for a query like ’computer science’, the search engine would rank the documents highest that have the two terms side-by-side. However, this additional information, i.e. the term proximity, makes the index larger and might make the search engine inefficient. This is where the concept of an inverted file is introduced. Most traditional search engines use this structure to represent their index with great results.

When representing such large amounts of data it is not feasible to list the words per document in an index. Instead an inverted index data structure is used which lists the documents per word. An inverted index is an index structure storing a mapping from words to their locations in a document or a set of documents, allowing full text search. This structure also optimizes the speed of the query as the query can look-up the word and find the documents containing it.

An inverted index has many variants but usually contains a reference to documents for each word and, possibly, also the position of the word in the document. If we have the set of texts

$T = \{\tau 0 = \text{i love you}, \tau 1 = \text{you love i}, \tau 2 = \text{love is blind}, \tau 3 = \text{blind justice}\}$,

Word/Term	index information
i	{ 0, 1 }
love	{ 0, 1, 2 }
you	{ 0, 1 }
is	{ 2 }
blind	{ 2, 3 }
justice	{ 3 }

Table 2. Inverted table

When searching for the words love and blind we get the result set {0, 1, 2} ∩ {2, 3} = {2}.

A full inverted index can be created from the same text set T by adding the local word number giving a full inverted index as seen in Table

Word/Term	index information
i	{ **(0, 0)**, (1, 2) }
love	{ **(0, 1)**, (1, 1), (2, 0) }
you	{ **(0, 2)**, (1, 0) }
is	{ (2, 1) }
blind	{ (2, 2), (3, 0) }
justice	{ (3, 1) }

Table 3. Full Inverted Table

When searching for the phrase i love you we get hits for both τ0 and τ1, but if we use the positioning we will only get τ0 as seen in bold in Table.

The index might also include details such as:

Position of a word

Position of the starting character

Term weights

Term frequency

- **Clustering**

Another possible way of indexing or improving an index, is to arrange documents into clusters, or categories. In this way, the documents dealing with similar or the same subject would be placed in the same category. Using categorization, it would be possible to only index the terms that are descriptive for each category. This would

decrease the size of the term histograms and in turn also decrease the size of the index. Using a clustered index can hopefully give us a more detailed index, which In turn would give more precise and relevant search results. For example, if a query for 'computer science' was submitted, the search engine would try and predict in which of the predefined categories the search terms are a part of. Then the search engine could search within these categories and return the results categorized and by relevance. However, this categorization comes with added computational effort since the clustering requires additional information stored in the inverted file structure and not to mention the calculation of the clusters themselves.

Essentially, the clustering could be yet another detail in our inverted file as mentioned above. Using steering as additional information in our index, the index would be more specific and contain the following details for each indexed term:

- List of documents where the term occurs
- The term weight
- Position within each document
- List of categories the word belongs to

As mentioned above, the indexer has to calculate the context for each document and decide which category, or even categories, it belongs to. Also, the categories that documents should adhere to must either be pre-calculated, in order to make this procedure faster, or they could be calculated in real-time.

Calculating the categories real-time is computationally expensive. This is because if a document is added to the index and does not match any of the already calculated categories, a new category is created. When a new category is added, all the already indexed documents need to be checked to see if they match the new category. If documents have been moved to new categories, all categories need to be recalculated. There are ways of doing such add ins more effectively, but it is very complex and still time consuming. Therefore, the predefined categories seem like the best way to go. However, this also comes at a price.

In order to predefine the categories, the indexer must be given a training-set to use for its categorization this training-set must be descriptive enough to be able to define

all the categories of the downloaded documents. Such datasets are hard to come by, and even harder to create so this is not an easy task.

Clustering Algorithms

Several algorithms have been developed and used to categorize text documents, some of which are listed below:

- Latent Semantic Indexing (LSI)
- Independent Component Analysis (ICA)
- Probabilistic Latent Semantic Indexing (PLSI)
- Bisecting k-means
- Spherical k-means
- K-Nearest Neighbor (kNN)
- Bayes-Classifier
- Principal Component Analysis (PCA)
- Artificial Neural Network
- Non-negative Matric Factorization (NMF)

Also, we found another algorithm, Frequent Term-based Clustering (FTC), promising but not many articles discuss its effectiveness or performance. The most common algorithms are probably LSI, PLSI, ICA and Bisecting k-means.

(c) Ranking

Whenever a user submits a query to a search engine, the engine returns a sorted list of results. This list is what the ranking algorithm in the search engine sees as the most relevant results, the first result being the most relevant one etc. In today's search engines these ranking algorithms can differ very much in design and performance. Any user familiar with the use of search engines knows that the results to a given query are rarely the same for the most popular search engines. In fact, these lists can be very different. This is in part due to which pages the search engines have indexed, but also due to the different underlying ranking algorithms.

Generally, ranking a web page is not as easy as it seems. For example, if a ranking algorithm is based solely on word matching and a user submits a query using very common keywords such as sports or movies, the algorithm ranks all pages containing these keywords equally and thus, possibly, gives the user a lot of useless results in random order. A more sophisticated algorithm would try and determine the relevance of the pages containing the keywords and rank them accordingly. As the number of web pages and other data on the Internet increases, returning relevant results to search queries becomes more difficult. Much effort has been put into the development of ranking algorithms in order to return more relevant results to the user.

The number of documents in the indices has been increasing by many orders of magnitude, but the user's ability to look at documents has not. People are still only willing to look at the first few tens of results.

A user is less likely to continue using a search engine which returns few relevant results within the first tens of the results. There are mainly three strategies in practice today when it comes to ranking search results. Namely, Link Analysis, Vector Space Model and Relevance Feedback. In the following subsection these strategies will be briefly introduced.

- **Link Analysis**

Link analysis in general is used to try and find a link between two subjects. For example, link analysis is used in law enforcement when a criminal's bank records, telephone calls etc. are investigated to try and find evidence of his or her crime. Banks and insurance companies also use this kind of link analysis to try and detect fraud. Within the field of search engines, the link analysis strategy to ranking is based on how the pages on the Internet link to each other. The assumption is that pages regarding a specific subject will, with good probability, link to other pages on similar or the same subject. This seems like a very reasonable assumption and has worked quite well in practice, e.g. Google uses this approach.

An example of ranking algorithm based on link analysis is Google's own Page Rank algorithm [23]... The following quote taken from Google's Technology Web page explains the essence of the Page Rank algorithm:

Page Rank relies on the uniquely democratic nature of the web by using its vast link structure as an indicator of an individual page's value. In essence, Google interprets a link from page A to page B as a vote, by page A, for page B. But, Google looks at more than the sheer volume of votes, or links a page receives; it also analyzes the page that casts the vote. Votes cast by pages that are themselves "important" weigh more heavily and help to make other pages "important".

So basically, a page's rank in Google's search results is higher if many, preferably important, pages link to that page. The higher the Page Rank, the more relevant the page is (according to Google). The mathematics behind the Page Rank algorithm is quite impressive but will not be explained here. For a good explanation of the mathematics and other design issues of the algorithm see [24].

Another example of a ranking algorithm using link analysis is the HITS algorithm. The HITS algorithm utilizes the link structure of the Internet like the Page Rank algorithm. The HITS algorithm is based on hubs and authorities, i.e. the algorithm calculates two values for each query, a hub value and an authority value. The authority value estimates the content of the page while the hub value estimates the value of its links to other pages.

- **Vector Space Model**

Ranking using Vector Space Model (VSM) is very simple. Basically, each document in VSM is represented as a column in a term-document matrix. Each row in the term-document matrix represents a term. The value at index $td_{i,j}$ says how many times term i occurs in document j. For example:

$$\mathbf{TD} = \begin{bmatrix} t_{1,1} & \dots & t_{1,m} \\ \vdots & \ddots & \vdots \\ t_{n,1} & \dots & t_{n,m} \end{bmatrix}$$

Where the column vector di is called the term-vector for document i. The term-vectors are often in very high dimensions as each term represents a single dimension, i.e. the richer the vocabulary, the higher the term-vector dimensions.

When ranking documents using VSM, the Cosine Similarity Measure calculates the angle between two vectors in the above matrix. The closer the angle is to zero, the more similar the two documents are. Therefore, when a query is submitted to a search engine, a term vector is constructed for that query in a similar way as it is done for documents. The best matches for that query are the documents with the highest similarity to the query vector.

Measuring similarity between two term-vectors is frequently done using the Cosine Similarity Measure. The cosine similarity measure between vector a and b is defined as the angle between the two vectors:

$$\cos\theta = \frac{\mathbf{a} \bullet \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|}$$

Where a • b is the dot product between the two vectors and |k| refers to the length of the vector. The similarity is found by looking at the angle between the two vectors. The smaller the angle, the greater the similarity between the vectors. Looking at figure, it is easy to see that the smaller the angle θ, the more similar the vectors |a| and |b| are. When θ = 0 the vectors are identical.

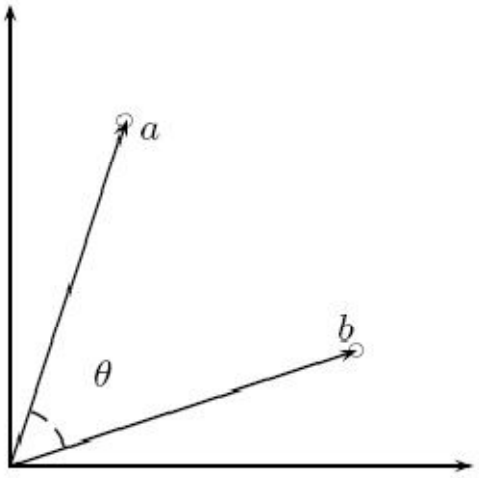


Fig no .5 Cosine similarity

The Vector Space Model and the calculation and usage of the Cosine Similarity Measure along with other similarity measures.

- **Relevance Feedback**

Relevance feedback is used to try and improve the relevance of the results returned by a search engine, where the search engine's original ranking usually follows one of the above mentioned schemes. The idea is that the search engine can learn which results are relevant for a given query. However, the machine has to get some feedback data in order to improve its results. Feedback information is used to either adjust the weights in a given query and/or add terms to the query to make it more specific. There are mainly two types of relevance feedback used, namely, *explicit feedback* and *implicit feedback*. Explicit feedback is where a user tells the search engine explicitly what results is relevant and which are not. Implicit feedback is where the search engine "monitors" which links a user follows for the given query. The search engine then makes the assumption that the links followed are more relevant than the others and in that way adjusts that page's rank in subsequent, similar queries.

It is intuitively clear that explicit feedback can be very useful to improve ranking results, given that the users are honest and consistent in their evaluations. In [25], Patil et. al. introduce a tool that can be used to get explicit feedback from users. Implicit feedback is not so intuitive since one can visit 10 search results before finding any useful result and therefore the other nine results should not get a higher ranking for the query. However, in [26], Jung et. al. found that considering all "click-data" in a search session has the potential to increase recall and precision of the queries. Also, in [27] Rohini and Ambati found that using implicit relevance feedback based on search engine logs and user profiles gave improved precision results.

2.1.4 Performance Measures

(a) Precision

The proportion of retrieved and relevant documents to all the documents retrieved:

$$\frac{\text{Total Number of Relevant Retrieved Documents}}{\text{Total Number of Retrieved Documents}}$$

A perfect **Precision** score of 1.0 means that every result retrieved by a search was relevant (but says nothing about whether all relevant documents were retrieved)

(b) Recall

The proportion of relevant documents that are retrieved, out of all relevant documents available:

$$\frac{\text{Total Number of Relevant Retrieved Documents}}{\text{Total Number of Relevant Documents}}$$

A perfect **Recall** score of 1.0 means that all relevant documents were retrieved by the search (but says nothing about how many irrelevant documents were also retrieved).

Both the measures are used to measure the accuracy of a system's ability to retrieve documents with respect to a given query. Ideally, you would like to achieve both high recall and high precision. In reality, you must strike a compromise. Indexing terms that are specific yields higher precision at the expense of recall. Indexing terms that are broad yields higher recall at the cost of precision. For this reason, an IR system's effectiveness is measured by the precision parameter at various recall levels.

(c) Some Other Measures

There are a number of more advanced and specific types of precision and recall measures that are used as modern evaluation measures. [9]

- ***Fallout*** is a measure of how quickly precision drops as recall is increased. Fallout is defined as the probability to find an irrelevant among the retrieved documents:

$$\frac{\text{Total Number of Irrelevant Retrieved Documents}}{\text{Total Number of Retrieved Documents}}$$

- **R-*precision*** is the precision at R where R is the number of relevant documents in the collection for the query. It is the precision after R retrieved documents, where R is the number of relevant documents that exists for that query. An R-precision of 1.0 is equivalent to perfect relevance ranking and perfect recall. However, a typical value of R-precision which is far below 1.0 does not indicate the actual value of
- **Recall** (since some of the relevant documents may be present in the hit-list beyond point R).
- ***Initial precision*** is the precision at recall 0% in the interpolated precision-recall graph. It is an indication of relevance ranking of the top few hits. Similarly, one can define a *final precision* that is the precision at 100% recall. Final precision indicates how far down one need to go in the hit-list to find all relevant documents.
- ***Precision** at 0.5 Recall* is the precision after half the relevant documents have been retrieved.
- ***Average Precision*** is the average of precision scores at every relevant document in the retrieved set.
- ***Recall** (1000)* is the recall after 1000 retrieved documents. This is more practical than true recall over all documents since modern systems can return a huge number of results

2.2 Basics of Information Theory

2.2.1 Information Theory

Information theory is a branch of applied mathematics and electrical engineering involving the quantification of information. Historically, information theory was developed by Claude E. Shannon to find fundamental limits on compressing and reliably storing and communicating data. Since its inception it has broadened to find applications in many other areas, including statistical inference, natural language processing, cryptography generally, networks other than communication networks — as in neurobiology[1] the evolution[2] and function[3] of molecular codes, model selection[4] in ecology, thermal physics,[5] quantum computing, plagiarism detection[6] and other forms of data analysis.[7]

A key measure of information in the theory is known as entropy, which is usually expressed by the average number of bits needed for storage or communication. Intuitively, entropy quantifies the uncertainty involved when encountering a random variable. For example, a fair coin flip (2 equally likely outcomes) will have less entropy than a roll of a die (6 equally likely outcomes).

Applications of fundamental topics of information theory include lossless data compression (e.g. ZIP files), loss data compression (e.g. MP3s), and channel coding (e.g. for DSL lines). The field is at the intersection of mathematics, statistics, computer science, physics, neurobiology, and electrical engineering. Its impact has been crucial to the success of the Voyager missions to deep space, the invention of the compact disc, the feasibility of mobile phones, the development of the Internet, the study of linguistics and of human perception, the understanding of black holes, and numerous other fields [citation needed]. Important sub-fields of information theory are source coding, channel coding, algorithmic complexity theory, algorithmic information theory, and measures of information.

- **Shannon–Hartley Theorem**

  In information theory, the Shannon–Hartley theorem is an application of the noisy channel coding theorem to the archetypal case of a continuous-time analog communications channel subject to Gaussian noise. The theorem establishes Shannon's channel capacity for such a communication link, a bound on the

maximum amount of error-free digital data (that is, information) that can be transmitted with a specified bandwidth in the presence of the noise interference, under the assumption that the signal power is bounded and the Gaussian noise process is characterized by a known power or power spectral density. The law is named after Claude Shannon and Ralph Hartley.

Considering all possible multi-level and multi-phase encoding techniques, the Shannon–Hartley theorem states that the channel capacity C, meaning the theoretical tightest upper bound on the information rate (excluding error correcting codes) of clean (or arbitrarily low bit error rate) data that can be sent with a given average signal power S through an analog communication channel subject to additive white Gaussian noise of power N, is:

$$C = B \log_2 \left(1 + \frac{S}{N}\right)$$

Where

C is the channel capacity in bits per second;

B is the bandwidth of the channel in hertz (pass band bandwidth in case of a modulated signal);

S is the total received signal power over the bandwidth (in case of a modulated signal, often denoted C, i.e. modulated carrier), measured in watt or volt2;

N is the total noise or interference power over the bandwidth, measured in watt or volt2; and

S/N is the signal-to-noise ratio (SNR) or the carrier-to-noise ratio (CNR) of the communication signal to the Gaussian noise interference expressed as a linear power ratio (not as logarithmic decibels

2.2.2 Entropy and Co-information

- **Entropy**

  The entropy, H, of a discrete random variable X is a measure of the amount of uncertainty associated with the value of X.

Suppose one transmits 1000 bits (0s and 1s). If these bits are known ahead of transmission (to be a certain value with absolute probability), logic dictates that no information has been transmitted. If, however, each is equally and independently likely to be 0 or 1, 1000 bits (in the information theoretic sense) have been transmitted. Between these two extremes, information can be quantified as follows. If {X}, is the set of all messages x that X could be, and p(x) is the probability of X given x, then the entropy of X is defined

$$H(X) = \mathbb{E}_X[I(x)] = -\sum_{x \in \mathbb{X}} p(x) \log p(x).$$

Here, I(x) is the self-information, which is the entropy contribution of an individual message, and {E}{X} is the expected value.) An important property of entropy is that it is maximized when all the messages in the message space are equiprobable p(x) = 1 / n,—i.e., most unpredictable—in which case $H(X) = \log_n$. The special case of information entropy for a random variable with two outcomes is the binary entropy function:

$$H_{\mathrm{b}}(p) = -p \log p - (1-p) \log (1-p).$$

- **Co-information**

The concept of co-information is a natural one to use in attempting to build a theory of machine learning, just as the concept of 'grouping of the variables' the co- information theory define as a part of probalistic method, entropy and mutual information. Both represent degrees of "hanging togetherness" amongst elements. We argue that the two concepts are the same. They arise from intuitions (both from mathematics and from neuroscience) those patterns (regularities, symmetries, dependencies, redundancies) in the world, can be captured in structured groupings of variables. In this context, the memories embedded in a group of variables are the patterns that are statistically more likely to appear in this group. The overall propensity of a group to produce patterns together is what we mean by the co-information: it is a measure of 'groupness' [3].

2.3 Contextual Retrieval on Web

Contextual retrieval is one of the major long term challenges in information retrieval. Contextual retrieval is defined as combine search technologies and knowledge about query and user context into a single framework in order to provide the most 'appropriate' answer for a user's information needs". The use of context in information retrieval is not a new idea. Jing et al. [33] use context as a basis of measuring the semantic distances between words. During indexing, the context of terms in documents is generated and stored in vector form. During retrieval, the context of a term in a query is generated and is used to measure the semantic distance between itself and candidate morphological variants in documents. Mutual information of terms is used to match related terms during the calculation of context distance. Billhardt et al. [34] propose a context-based vector space model for information retrieval. After the term-document matrix has been constructed, it is used is a basis for generating a term context matrix where each column is considered a semantic description of a term. This term context matrix is then combined with the document vectors from the term-document matrix to transform it into the final document context vector used for retrieval. The WEBSOM [35] system is an example of another way in which context has been used for information retrieval. It uses a two level Kohonen's self-organizing map approach to group words and documents of contextual similarity. Context in WEBSOM is limited to the terms that occur direct either sides of the term in question. IntelliZap [36] is a context-based web search engine that requires the user to select a key word in the context of some text. The approach makes effective use of the contextual information in the immediate vicinity of the keywords selected, so that retrieval precision can be improved. Inquires [37, 38] is another web search engine that uses contextual information to improve search results. A user must specify some contextual information, considered as preferences, pertaining to the query. This context (preferences) provides a high-level description of the users information need and ultimately control the search strategy used by the system. Hyperlink information can be a very valuable source

of evidence for web information retrieval and it is either based on a set of retrieved documents during retrieval or on a global analysis of the entire document collection during indexing. Kleinberg [39] illustrates how hyperlink information in web pages can be used for web search when using a set of retrieved documents. An approach that also uses the characteristics of link information from a set of retrieved documents for topic distillation is presented by Amitay et al. [40]. PageRank, as proposed by Brin et al. [41], is hyperlink-based retrieval algorithm that calculates document scores by considering the entire hyperlink connected graph represented by all the links in the entire document collection. It uses link information to model user behavior by calculating the probability that a user will eventually visit a certain page. This probability or Page Rank of a page is used to prioritize its ranking during retrieval.

The model with the most similar form of ours is [44], though it uses traditional query expansion to determine context of query**.** Another closely related work [43], implicitly deduce context using three different algorithms. Finally [42], offers Term context model as a new tool for accessing term presence in a document.

Chapter 3

The Algorithmic Framework

In this chapter we give the details of the Context Detection Algorithm designed to extract contextual evidence from the user query. The algorithm is based on an information-theoretic measure: the co-information and is used to generate a context coverage list given a multi-term query.

3.1 Multi-term Query

Multi term query of keywords plays a quintessential role in the Web search paradigm. Recent studies claim that queries involving one or two keywords are most common in Web searches. While most Web Search engines perform very well for a single-keyword query, their precision is not good for query involving two or more keywords because the search results usually contain a large number of pages with weak relevance. Also, the users have a well-defined query re-formulation behavior, i.e., most multiple term queries include more than one context and users usually reformulate their queries by context instead of terms. A context is usually included as a sub-query in a user's query and it has strong impacts on the quality of search results.

3.2 Using Snippets

A query typically contains only a few terms, which provide limited information. One straightforward method is to submit a query to a search engine to get the top ranked search pages. Those retrieved results provide some richer information about the query. In other words, we call the retrieved results of query as the local information of this query. Meanwhile, a query has its global information, based on the whole corpus, to provide more information. However, the global based approach can cause high computational complexity and it was shown in [3] that a local based approach outperforms the global

based approach. So in this research, the top ranked search results are utilized to enrich the query.

In the local analysis, we select significant characteristics from each (retrieved and relevant) document and name them 'features', i.e., given the search results for a query, we need to decide what features should be extracted from the search engine to construct the enrichment. Generally, three kinds of features are considered: the title of a page, the snippet generated by the search engine, and the full plain text of a page. We use the top N ranked snippet retrieved by search engine as the local search result of query. A Snippet is a short fragment of text extracted from the document content [18].

3.3 Co-information metric

Co-information measures the information that two discrete random variables share: it measures how much knowing one of these variables reduces our uncertainty about the other. As for a sub-query generated by the original query, the Co-Information between its terms can be used to measure the information bound up in those terms. The more information bound up one sub-query has, the higher possibility to be a topic it has. Mathematically this information-theoretic measure is defined as [3]

$$I(x_{1i}; x_{2j}; \ldots; x_{nl}) = \sum_{i,j,\ldots,l} p(x_{1i}, x_{2j}, \ldots, x_{nl}) \left(\sum_{|v'| \subseteq |T'|} (-1)^{|v'| - |T'|} \log (p(T')) \right).$$

where v' = (x_{1i}; x_{2j}; …; x_{nl}) and T' is any subset of v'. v' means the number of elements in v', and the same with T' . In this way, for n events x_{1i}, x_{2j}, …, x_{nl}, we define the co-information, CI, between them I (x_{1i} ; x_{2j} ; …; x_{nl}) by

$$I(x_{1i}; x_{2j}; \ldots; x_{nl}) = \sum_{|v'| \subseteq |T'|} (-1)^{|v'| - |T'|} \log (p(T')).$$

We utilize the Co-Information metric to measure the degree of one sub-query being a topic. In our approach, the probability space of one query is built based on the local

results of all sub-queries. Given a query Q = t_1 t_2 …t_n, where t_i (1<= i <=n) is the i^{th} term of Q, we obtain the context coverage list as described in the following algorithm:

3.4 Context Detection Algorithm

Given a query Q = t_1 t_2 …t_n, where t_i (1<= i <=n) is the i^{th} term of Q, the context coverage list is obtained as described in the following algorithm [3]:

Step 1: Get the set of all sub-queries, SQ = {s_{qk}}, 1<=k<=2^{n+1}-2, where 1<=k<=2^{n+1}-2 is the number of all sub-queries and 2^{n-2} = S1 ^{n} +S2 ^{n} +…+Sn-1 ^{n}

Step 2: Enrich each sub-query by submitting it into search engine and get the top N ranked snippets. In this way, we can get (2^{n}-2). N* snippets for query q. The snippet set is defined as S(Q) = {s_{ni}}, 1<=i<=(2^{n}-2). N*, where s_{ni} is the ith snippet. N* = min(N, N'), where N' is the actual number of retrieved snippets by search engine for each query.

Step 3: The probability of sub-query in S (Q) is defined as

$$p(sq_k) = \sum_{i=1}^{(2n-2).N^*} (sq_k \text{ can be found in } sn_i)/(2^n - 2).N^*$$

p (s_{qk}) is the probability of occurrence of s_{qk} in collection.

Step 4: As for s_{qk}, we get its sub-query set, defined as S_{Qk} = { s_{qk1}..s_{qk2},….s_{qk2} ^{k} $_{-2}$}.

$$I(sq_k) = \sum_{|sq_k| - |sq_{kj}|} (-1)^{|sq_k| - |sq_{kj}|} \log (p(sq_{kj})).$$

Step 5: Order all sub-queries in a descendant value of Co-information and split it into two parts by the threshold of zero. In other words, the first part includes the sub queries with positive CI and the last part includes the sub-queries with negative CI.

Step 6: Finally, the list of context words we detected from S_Q is defined as: Context Coverage List, CL = { C_1, C_2, …., C_i,…C_M }, where C_i is one sub-query which has positive CI and M is the total number, and I(C_i) >= 0, I(C_i)>=I(C_{i-1}), 1<= i<=M.

Chapter 4.

Experimental Setup and Analysis

In this chapter we give the details of implementation. An application is built that gives the context coverage list as output which can be used for query enrichment to yield remarkable improvement in the performance.

4 .1 Implementation Detail

We developed our application in C# which is a high performance language for technical computing and programming in an easy to use environment where problems and solution are expressed in familiar mathematical notation.

The create graphical user interface in C# which provides user a interface by which put own query and find the related document from the database and find the context of the related to user query and provide the coverage list from the database .

The data is use a static database because the corpus data base is very difficult to fetch without purchasing but we have try to fetch with the help of Google API, yahoo API and that is not feasible. So we have created the data base and store the document. So this application is a prototype.

The Graphical User Interface

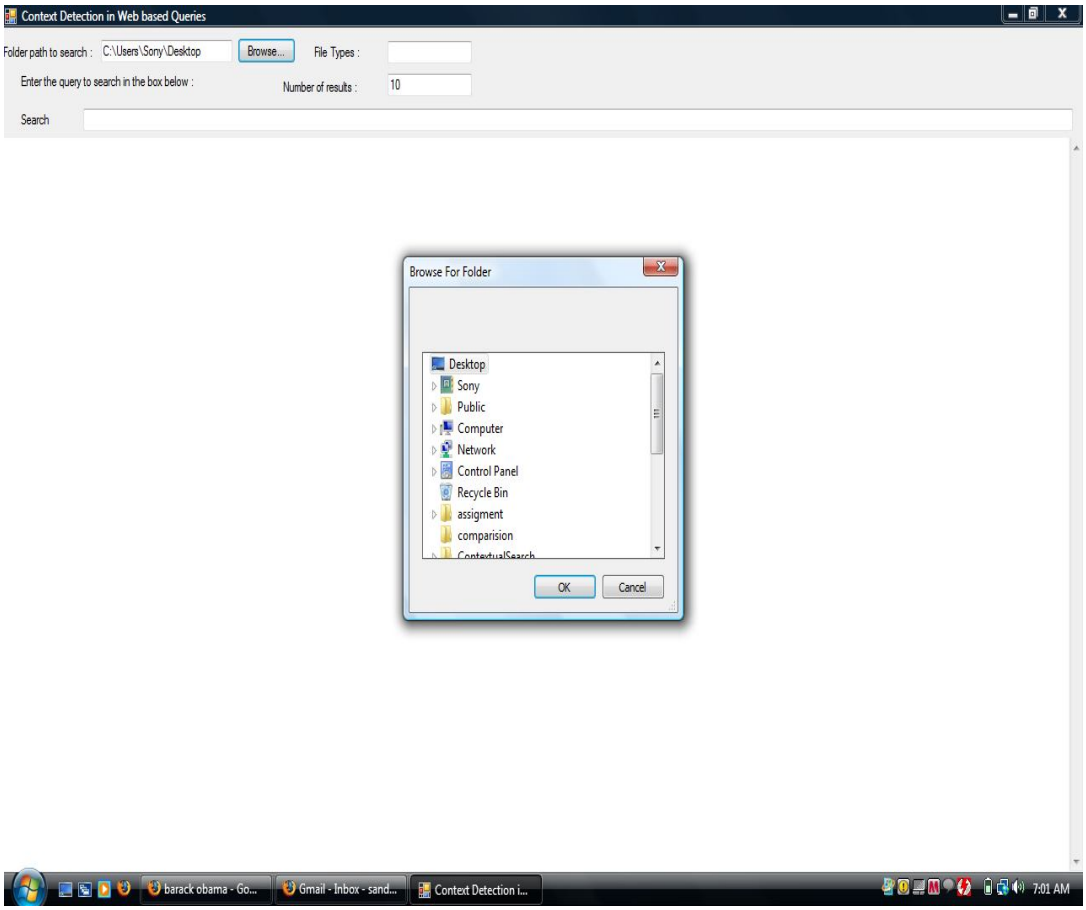


Figure 5. Graphical user interface

User insert file type in the option field and put own query in query field.

4.2 Sample Snippets

To detect the context we have considered some queries and snippets generated specific to the query are as follows:

Example Queries:

(a) What is bank loan?

(b) Bank mortgage?

(c) Borrower and finance?

Sample Snippet 1

Banks (0) provide credit money against *property* **(6)**…….some **mortgage (10)** ……..……all eligible to apply for the *loan* **(18).** It enables the borrower to apply for finance against a fixed asset, from **bank (28)**. The maximum amount of ……..... Repayment is done through Equated Monthly Instalments or EMI made to the **bank (85)**. A wide range of……. **mortgage (97)** types are available.

Sample Snippet 2

Banks (0) provide credit money against *property* **(6)**…….some **mortgage (10)** ……..……all eligible to apply for the *loan* **(18).** It enables the borrower to apply for finance against a fixed asset, from **bank (28)**. The maximum amount of ……..... Repayment is done through Equated Monthly Instalments or EMI made to the **bank (85)**. A wide range of……. **mortgage (97)** types are available.

Sample Snippet 3

Banks (0) provide credit money against *property* **(6)**…….some **mortgage (10)** ……..……all eligible to apply for the *loan* **(18).** It enables the borrower to apply for finance against a fixed asset, from **bank (28)**. The maximum amount of ……..... Repayment is done through Equated Monthly Instalments or EMI made to the **bank (85)**. A wide range of……. **mortgage (97)** types are available.

Chapter 5

Conclusion

Conclusion

We implemented a novel context detection algorithm given a user's query. The aim was to use "Context as a query" and treat the context as a background for possible refinement of query.

Possible future directions can be towards making this technique easily pluggable into existing systems and also finding alternatives other than Co-information metric for the detection of context.

References

[1] MPS, Bhatia, & Akshi, K Khalid. “A Primer on the Web Information Retrieval Paradigm”, *Journal of Theoretical and Applied Information Technology (JATIT)*, 2008, 4, (7), pp 657-662.

[2] MPS, Bhatia, & Akshi, K Khalid “The Context-driven Generation of Web Search”, *Proceedings of CISTM*, 2008, pp 281-287.

[3] MPS, Bhatia, & Akshi, K Khalid “Contextual Proximity Based Term-Weighting for improved Web Information Retrieval”, *Proceedings of KSEM 2007, Lecture notes of AI-4798*, Springer, 2007, pp 267-278.

[4] H. Billhardt, D. Borrajo, V .Maojo, “A context vector model for information retrieval”. J. Am. Soc. Inf. Sci. Technol. 53(3), 236–249 (2002).

[5] L. Finkelstein, E.Gabrilovich, Y.Matias, E.Rivlin, Z.Solan, G. Wolfman, E.Ruppin.: “Placing Search in context: the concept revisited”. In: Proceedings of the 10th International World Wide Web Conference, pp. 406–414 (2001)

[6] Stephen Robertson topic” Understanding Inverse Document Frequency: On theoretical arguments for IDF” Microsoft Research Reprinted from: Journal of Documentation 60 no. 5, pp 503–520

[7] M. Kobayashi, K.Takeda: “Information Retrieval on the Web”. ACM Computing Surveys 32(2) (2000)

[8] Monika Henzinger, “Information Retrieval on the Web”, 39th Annual Symposium on Foundations of Computer Science (FOCS'98), Palo Alto, CA.

[9] Orland Hoeber and Xue Dong Yang” Interactive Web Information Retrieval Using WordBars” Department of Computer Science University of Regina Regina, Saskatchewan, Canada S4S 0A2 IEEE-2006

[10] A. Z. Broder. Min-wise independent permutations. In Journal of Computer and System Sciences, volume 60, pages 630–659, 2000.

[11] I.A. Witten, A.Moffat, and T.C. Bell. Managing Gigabytes: Compressing and Indexing Documents and Images. Morgan Kauffam, 1999.

[12] S. Chakrabarti. Mining the Web: Analysis of Hypertext and Semi Structured Data Mining the Web: Analysis of Hypertext and Semi Structured Data. Morgan Kaufmann, 2003.

[13] Hearst, M. A. (2000). Next generation web search: Setting our sites. IEEE Data Engineering Bulletin, Special issue on Next Generation Web Search, Luis Gravano (Ed.)

[14] S. Brin and L. Page. The anatomy of a large-scale hyper textual Web search engine. In Proceedings of 7th International World Wide Web Conference, Brisbane, Australia, pages 107–117, 1998.

[15] J. Kleinberg. Authoritative sources in a hyper-linked environment. In Proceedings of 9th ACM-SIAM Symposium on Discrete Algorithms, San Francisco, U.S., pages 668–677, 1998.

[16] R. Lempel and S. Moran. The stochastic approach for link-structure analysis (SALSA) and the TKC effect. In Proceedings of the 9th International World Wide Web Conference, Amsterdam, The Netherlands, pages 387–401,2000.

[17] K. Bharat and M. Henzinger. Improved algorithms for topic distillation in a hyper-linked environment. In Research and Development in Information Retrieval, pages 104–111, 1998.

[18] Fabrizio Sebastiani. Machine learning in automated text categorization. ACM Comput. Surv., 34(1):1–47, 2002.

[19] Ricarco Baeza-Yates and Berthier Ribeiro-Neto. Modern Information Retrieval. Addison-Wesley, 1999.

[20] W.B.Frakes and R. Baeza-Yates. Information Retrieval: Data Structures and Algorithms. Prentice Hall, 1992.

[21] D. Hull and G. Grefenstette. A detailed analysis of english stemming algorithms. Technical report, Rank XEROX, 1996.

[22] Franca Debole and Fabrizio Sebastiani. “Supervised term weighting for automated text categorization”. In SAC ’03: Proceedings of the 2003 ACM symposium on Applied computing, New York, NY, USA, ACM Press, pages 784–788, 2003.

[23] Sergey Brin and Lawrence Page. The anatomy of a large-scale hyper textual Web search engine. Computer Networks and ISDN Systems, 30(1–7):107– 117, 1998.

[24] Amy N. Langville and Carl D. Meyer. Google’s PageRank and beyond: The science of search engine rankings. Princeton University Press, 2006.

[25] Stefan Wild, James Curry, and Anne Dougherty. Improving non-negative matrix factorizations through structured initialization. Pattern Recognition, 37(11):2217–2232, 2004.

[26] A. Kehagias, V. Petridis, V. Kaburlasos, and P. Fragkou. A comparison of word- and sense-based text categorization using several classification algorithms, 2003

[27] Ambati Vamshi and U Rohini. Improving re-ranking of search results using collaborative filtering. Lecture Notes in Computer Science, 4182:205–216, 2006.

[28] Stephen Robertson "Understanding Inverse Document Frequency: On theoretical arguments for IDF" Microsoft Research 7 JJ Thomson Avenue Cambridge CB3 0FBUK (and City University, London, UK) Journal of Documentation 60 no. 5, pp 503–520

[29] R.R. Joshi and Y.A. Aslandogan. Concept-based web search using domain prediction and parallel query expansion. Proceedings of the 2006 IEEE International Conference on Information Reuse and Integration (IEEE Cat No.06EX1467), page 6 pp., 2007.

[30] J. Zakos, B.Verma,: A Novel Context-based Technique for Web Information Retrieval World Wide Web 9(4), 485–503 (December 2006).

[31] A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. In Proceedings of 14th International World Wide Web Conference, pages 902–903, Chiba, Japan, 2005.

[32] R. Ali and M.M.S. Beg. A framework for evaluating web search systems. WSEAS Transactions on Systems, 6(2):257–264, 2007.

[33] H. Jing, E Tzoukermann: Information retrieval based on context distance and morphology. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in information Retrieval, pp. 90–96. ACM Press, New York (1999)

[34] T. Honkela, S. Kaski, K.Lagus, T Kohonen,: WEBSOM — self-organizing maps of document collections. In: Proceedings of WSOM_97 (Workshop on Self-Organizing Maps), Espoo, Finland, pp. 310–315 (1997)

[35] E.Glover, S.Lawrence, M., Gordon, W.Birmingham, C Lee Giles,.: Web search your way. Commun. ACM 44(12), 97–102 (2001)23. Lawrence, S., Giles, C.: Context and page] analysis.

[36] S. Lawrence, C. Giles, Context and page analysis for improved web search. IEEE Internet Computing 2(4), 38–46 (1998).

[37] J.Kleinberg,: Authoritative sources in a hyperlinked environment. Journal of the ACM 46(5), 604–632 (1999)

[38] E. Amitay, D.Carmel, A.Darlow, R.Lempel, A.Soffer: Topic distillation with knowledge agents. In: Proceedings of the 11th Text Retrieval Conference (TREC-11), Gaithersburg, Maryland, USA (2002

[39] S.Brin, L.Page: The anatomy of a large-scale hyper-textual web search engine. In: Proceedings of the 7th WWW Conference, Brisbane, Australia, pp. 107–117 (1998a).

[40] J.Pickens, A.M. Farlance,: Term Context Models for Information Retrieval, ACM CIKM (2006)

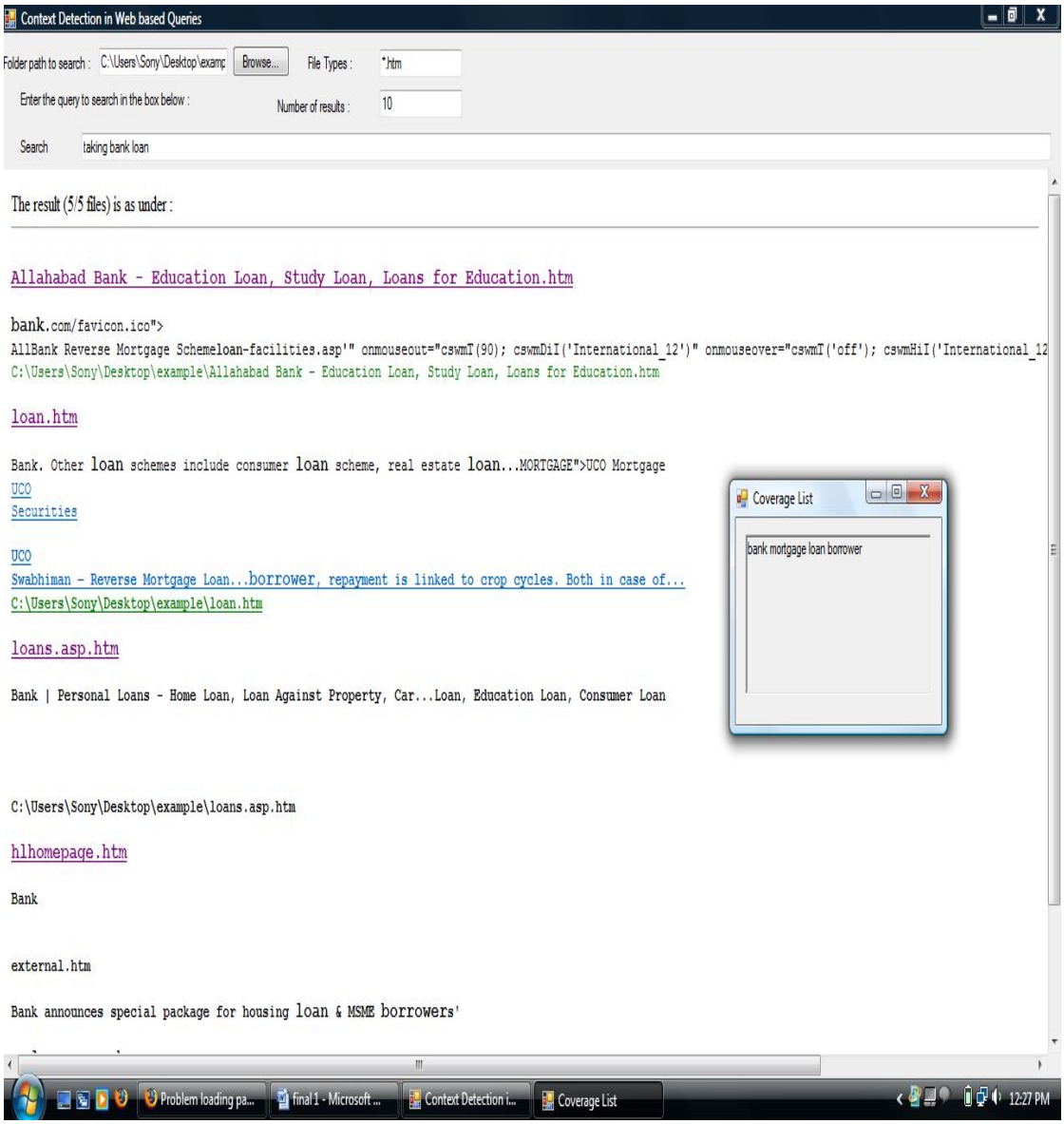
[41] S.Jonathan, et al.: Context Driven Ranking for the Web.

[42] J .Zakos, B.Verma,: A Novel Context-based Technique for Web Information Retrieval World Wide Web 9(4), 485–503 (December 2006)

[43] S. Birch. Statistical text modelling - towards modelling of matching problems.Master's thesis, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK- 2800 Kgs. Lyngby, 2003.

Appendix A: Screen-Shots

Q.1Taking Bank Loan



Q.2 About Bollywood Music