International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 – 5161

Available at http://inpressco.com/category/ijcet

Research Article

A Novel Approach for Information retrieval using Web based search engine

Vipul Narayan[1*], R.D.S.Yadav[2], R.K.Mehta[3], Mahendra Rai[4], Musheer Ahmed [5], Rahul Maurya[6], Abhishek Kanujiya [7] and Dharmpal [8]

[1,5,6,7,8]Department of C.SE, N.D.U.A.T.Faizabad India
[2]M.C.A.E.T. NDUAT, Faizabad India
[3]Deptt. of IDE, NDUAT, Faizabad India
[4]Deptt. of SWCE., NDUAT, Faizabad India

Accepted 15 June 2017, Available online 27 June 2017, **Vol.7, No.3 (June 2017)**

Abstract

Nowadays, people frequently use search engines in order to find the information they need on the web. However, usually web search engines return web page references in a global ranking making it difficult for the users to browse different topics captured in the result set and thus making it difficult to find quickly the desired web pages. There is a need for special computational systems that will discover knowledge in these web search results, providing the user with the possibility to browse different topics contained in a given result set. In this paper, we focus on the problem of determining different thematic groups on web search engine results that existing web search engines provide. We propose a novel system that exploits a set of reformulation strategies so as to help users gain more relevant results to their desired query. It furthermore tries to determine, among the result set different topic groups, according to the various meanings of the provided query. The proposed method utilizes a number of semantic annotation techniques using Knowledge Bases, like Word Net and Wikipedia, in order to perceive the different senses of each query term. Finally, the method annotates the extracted topics using information derived from the clusters and presents them to the end user.

Keywords: *Information, search engine, web engine, query, world wide web etc.*

1. Introduction

1.1 Information Retrieval

Information retrieval is a field referenced with the analysis, organization, structure, searching, storage, and retrieval of information. The term “information” is very general, and information retrieval includes work on a wide range of types of information and a variety of applications related to search. The usual search scenario involves someone typing in a query to a search engine and receiving answers in the form of a list of documents in ranked order. Although searching the World Wide Web (web search) is by far the most common application involving information retrieval, search is also a crucial part of applications in corporations, government, and many other domains (Brill *et al*(2006)),Tsegay *et al* (2007)). A retrieval model is a proper representation of the process of matching a query and a document. It is the basis of the ranking algorithm that is used in a search engine to produce the ranked list of documents Berger *et al* (2000).

*Corresponding author: **Vipul Narayan**

1.2 Search Engine

A search engine is the real-world application of information retrieval techniques to extensive text collections. A web search engine is the obvious example, but as has been mentioned, search engines, can be found in many different applications, such as desktop search or enterprise search (Brill *et al* (2006)). Web search engines, such as Google and Yahoo!, 7 must be able to capture, or crawl, many terabytes of data, and then provide sub second response times to millions of queries submitted every day from around the world. Enterprise search engines for example, Autonomy must be able to process the large variety of information sources in a company and use company-specific knowledge as part of search and related tasks, such as data mining. Data mining refers to the automatic discovery of interesting structure in data and includes techniques such as clustering. Desktop search engines, such as the Microsoft Vista search feature, must be able to rapidly incorporate new documents, web pages, and email as the person creates or looks at them, as well as provide an intuitive interface for searching this very heterogeneous mix of information. In the beginning, with the least number of

sites on the Web, the use of a search engine was limited. To find information, the user could directly explore the source and if required could use the facility called "What's New" provided by the National Centre for Supercomputing. Metadata is information about a document that is not part of the text content, such the document type (e.g., email or web page), document structure, and other features, such as document length etc. (Dean *et al* (2003)), (Belew *et al* (2000)), Belkin *et al* (2008)).This paper is organised in following ways .In section I author discussed about introduction about the paper. In section II literature review by various authors. In section III architecture of web based search engine and query process is explained. In section IV crawls and feed methods for information retrieval is explained and in last section V conclusion and future work is explained.

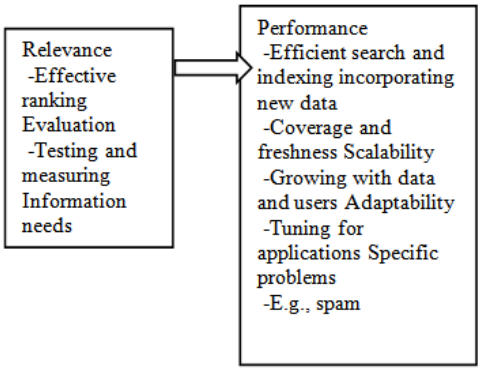

Fig. 1.1 Search engine design and the core information retrieval issues

2. Literature review

(Brill *et al* (2006)) discussed on the history of World Wide Web and its sizes, was provided wherein it was highlighted that the World Wide Web contains a huge amount of data (billions of web pages) in the form of electronic documents, distributed all over the world and still growing at an exponential rate. How will it be possible for the end users to retrieve the required data from the Web? As a result, there is a need for information retrieval tools which may help the end users to access the required information. Speed, precision and recall are the three main features to evaluate the performance of any classical information retrieval system. Precision is defined as the ratio of relevant documents to the number of retrieved documents, whereas recall is defined as the ratio of relevant documents that are retrieved to the total number of relevant documents i.e.

$$precision = \frac{number\ of\ relevant\ documents}{number\ of\ retrieved\ documents}$$

$$Recall = \frac{\text{number of relevant,retrieved documents}}{\text{Total number of relevant documents}}$$

The most popular information retrieval tools are Web directory, Search engine and Meta search engines.

In the web directory, web page's hyperlinks are represented hierarchically in a tree form. This representation is broken down into topics and subtopics and may have many levels determined on the basis of categorization of the topic. It is not a search engine and does not display lists of web pages based on keywords, instead it lists web sites by category and subcategory. The categorization is generally based on the entire web site somewhat than one page or a set of keywords. To classify the web pages and the presentations that are added to a web directory, involves the intervention of human editors. For representation of the web pages, it is easy to use the directory structure as by using this the user need not know what exactly he is looking for in order to get something meaningful. As the representation is broken down into topics and subtopics, the user can select a suitable category for a topic of his interest and can move down through the hierarchy, selecting the subcategory and thus narrowing the search at each level until he is offered a list of hyperlinks that are relevant to his topic. While traversing the directory structure downwards, the user moves towards more specific topics whereas traversing upwards he moves towards more general topics. Nowadays, directories are seeking the assistance of search engines to enhance what they find in their own directories. 'The100 Lists' web directory is an example which is completely human edited and is a friendly search engine that adheres to the World Wide Web Consortium (W3C) standards in an attempt to help lead the World Wide Web to its full potential. Another example of Web directories is 'Yahoo' and 'Open Directory Project (ODP).

(Dean *et al* (2003)) proposed terms Internet and World Wide Web are frequently used in everyday speech without much distinction between them. However, both are not one and the same. The Internet is a hardware and software infrastructure of world-wide data communication system that offers connectivity between computer systems while the Web is one of the services communicated via the Internet and is a collection of interconnected documents and other resources, linked through hyperlinks and URLs. In the beginning, with the least number of sites on the Web, the use of a search engine was limited. To find information, the user could directly explore the source and if required could use the facility called "What's New" provided by the National Centre for Supercomputing. This application provided a simple listing of the new websites that would pop up on a daily basis. "Jerry's Guide to the World Wide Web," was another tool which later became Yahoo. These manual search tools were slow. As the Web began to grow exponentially, both in the number of sites and the level of public interest, the users desperately needed tools to quickly sort, categorize, and search the ever-growing volume of information. Starting in 1994, a number of search engines were launched, including AltaVista, Excite, Infoseek, Inktomi, Lycos, and of course the

evergreen, Yahoo and Google. These search engines, mostly save a copy of the web pages in their central repository and then make appropriate indexes of them for later search/retrieval of information.

(Tsegay *et al* (2007)) said that search engine searches documents in its local storage for specified keywords provided by the user and returns a list of the documents containing the keywords. The documents returned may consist of web pages, images, information and other types of files. The first web robot called World Wide Web wanderer was developed by Matthew Gray in 1993. The aim behind its development was to create indexes which help to estimate the size of WWW. The first search engine called "Aliweb" was developed in November1993 but it did not depend on web robot to crawl the web pages rather it was notified by website administrator about the existence of an index file. The first true World Wide Web search engine was Jump Station developed in December 1993, which had all three essential features of a search engine namely crawling, indexing and searching but due to resource constraints it indexed only titles and headings found in web pages. In 1994, WebCrawler, the first publicly known full text crawler meta-search engine was developed by Brian Pinkerton, which later became a standard for all major search engines. After this, many popular search engines such as Excite, Infoseek, AltaVista, Yahoo and others were designed. Around 1998, Larry Page and Sergey Brin developed Google which later became the most popular search engine among all. The important feature of this search engine is that the results produced by it are ordered on the basis of PageRank. Apart from creating indexes and maintaining caches of web pages it also takes "snapshots" of other file types, which include PDF, Word documents, Excel spreadsheets, Flash SWF, plain text files, and so on.

A search engine generally consists of the following components:

• User Interface • Query Engine • Indexer 19 • Crawlers) • Repository

3. Architecture of web based search engine and query process

3.1 What is architecture?

Architecture is used to describe a system at a particular level of abstraction. Our search engine architecture is used to present high-level descriptions of the important components of the system and the relationships between them. It is not a code-level description, although some of the components do correspond to software modules in the Galago search engine and other systems. We use this architecture in this chapter and throughout the book to provide context for the discussion of specific techniques. Architecture is designed to ensure that a system will satisfy the application requirements or goals. The two primary goals of a search engine are:

• Effectiveness (quality): We want to be able to retrieve the most relevant set of documents possible for a query.
• Efficiency (speed): We want to process queries from users as quickly as possible. (Larkey *et al* (2003)), (Brill *et al* (2006)), (Castillo *et al* (2008)), (Azzopardi *et al* (2006)), (Dean *et al* (2003)).

3.2 Basic Building Blocks

Search engine components support two major functions, which we call the *indexing process* and the *query process*. The indexing process builds the structures that enable searching, and the query process uses those structures and a person's query to produce a ranked list of documents.

Figure 3.1 shows the high-level "building blocks" of the indexing process. These major components are text acquisition, text transformation, and index creation. The task of the text acquisition component is to identify and make available the documents that will be searched. Although in some cases this will involve simply using an existing collection, text acquisition will more often require building a collection by crawling or scanning the Web, a corporate intranet, a desktop, or other sources of information. The task of the text acquisition component is to identify and make available the documents that will be searched. Although in some cases this will involve simply using an existing collection, text acquisition will more often require building a collection by crawling or scanning the Web, a corporate intranet, a desktop, or other sources of information. In addition to passing documents to the next component in the indexing process, the text acquisition component creates a document data store, which contains the text and metadata for all the documents. Metadata is information about a document that is not part of the text content, such the document type (e.g., email or web page), document structure, and other features, such as document length.

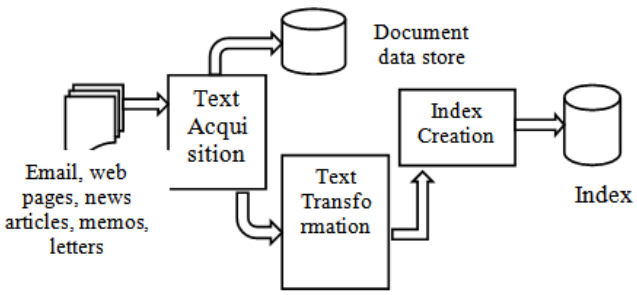

Fig. 3.1 The indexing process (Dean *et al* (2003))

Figure 3.2 shows the building blocks of the query process. The major components are user interaction, ranking, and evaluation. The user interaction

component provides the interface between the person doing the searching and the search engine. One task for this component is accepting the user's query and transforming it into index terms. Another task is to take the ranked list of documents from the search engine and organize it into the results shown to the user. This includes, for example, generating the snippets used to summarize documents. The document data store is one of the sources of information used in generating the results. Finally, this component also provides a range of techniques for refining the query so that it best represents the information needed (Allan *et al* (1996)), (Brill *et al* (2006)), (Berger *et al* (2000)), (Tsegay *et al* (2007)).

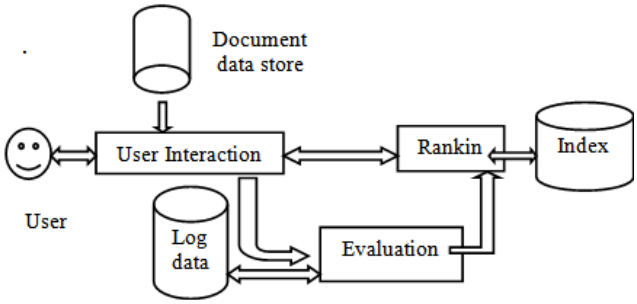

Fig. 3.2 The query process

4. Crawls and feeds method for information retrieval using web search engine

4.1 Deciding What to Search

"What should we search?" The simple answer is everything you possibly can. Every document answers at least one question (i.e., "Now where was that document again?"), although the best documents answer many more. Every time a search engine adds another document, the number of questions it can answer increases. On the other hand, adding many poor-quality documents increases the burden on the ranking process to find only the best documents to show to the user. Web search engines, however, show how successful search engines can be, even when they contain billions of low-quality documents with little useful content.

4.2 Crawling the Web

Finding and downloading web pages automatically is called crawling, and a program that downloads pages is called a web crawler. Crawling is also occasionally referred to as spidering, and a crawler is sometimes called a spider. There are some unique challenges to crawling web pages. The biggest problem is the sheer scale of the Web. There are at least tens of billions of pages on the Internet.

4.2.1 Retrieving Web Pages: Each web page on the Internet has its own unique uniform resource locator, or URL. Any URL used to describe a web page has three parts: the scheme, the hostname, and the resource name (Figure 3.1). Web pages are stored on web servers, which use a protocol called Hypertext Transfer Protocol, or HTTP, to exchange information with client software. Therefore, most URLs used on the Web start with the scheme http, indicating that the URL represents a resource that can be retrieved using HTTP. The hostname follows, which is the name of the computer that is running the web server that holds this web page. In the figure, the computer's name is www.cs.umass.edu, which is a computer in the University of Massachusetts Computer Science department. This URL refers to a page on that computer called /csinfo/people.html.

http:// www.cs.umass.edu/ csinfo/ people. html
http www. cs.umass .edu/csinfo/people. html
Scheme hostname resource

Fig. 4.1 A uniform resource locator (URL) splits into three parts

Web browsers and web crawlers are two different kinds of web clients, but both fetch web pages in the same way. First, the client program connects to a domain name system (DNS) server. The DNS server translates the hostname into an internet protocol (IP) address. The program then attempts to connect to a server computer with that IP address. By convention, requests for web pages are sent to port 80 unless specified otherwise in the URL. Once the connection is established, the client program sends an HTTP request to the web server to request a page. The most common HTTP request type is a GET request, for example:
GET /csinfo/people.html HTTP/1.0

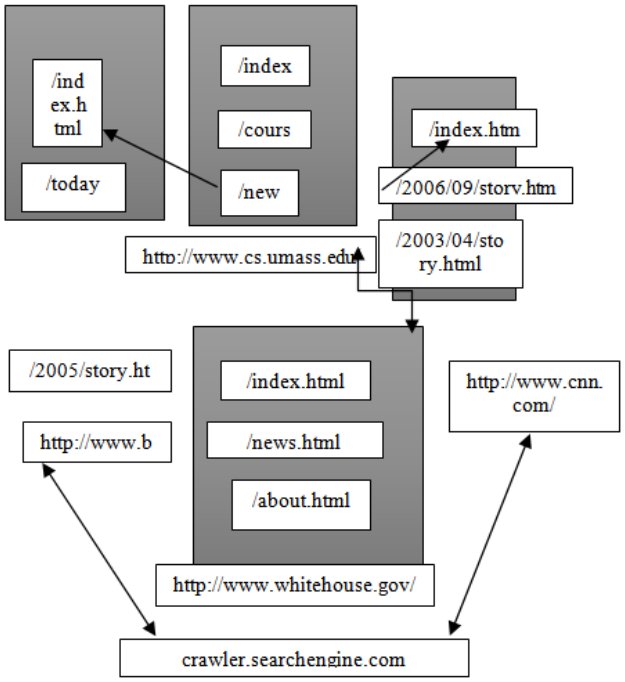

Fig. 4.1 Crawling the Web. The web crawler connects to web servers to find pages. Pages may link to other pages on the same server or on different servers

This simple request asks the server to send the page called /csinfo/people.html back to the client, using version 1.0 of the HTTP protocol specification. After sending a short header, the server sends the contents of that file back to the client. If the client wants more pages, it can send additional requests; otherwise, the client closes the connection. A client can also fetch web pages using POST requests. A POST request might be used when you click a button to purchase something or to edit a web page. This convention is useful if you are running a web crawler, since sending only GET requests helps make sure your crawler does not inadvertently order a product.

4.2.2 The Web Crawler: The crawler starts with a set of seeds, which are a set of URLs given to it as parameters. These seeds are added to a URL request queue. The crawler starts fetching pages from the request queue. Once a page is downloaded, it is parsed to find link tags that might contain other useful URLs to fetch. If the crawler finds a new URL that it has not seen before, it is added to the crawler's request queue, or frontier. The frontier may be a standard queue, or it may be ordered so that important page move to the front of the list. This process continues until the crawler either runs out of disk space to store pages or runs out of useful links to add to the request queue. If a crawler used only a single thread, it would not be very efficient. Notice that the web crawler spends a lot of its time waiting for responses: it waits for the DNS server response, then it waits for the connection to the web server to be acknowledged, and then it waits for the web page data to be sent from the server. During this waiting time, the CPU of the web crawler machine is idle and the network connection is unused. To reduce this inefficiency, web crawlers use threads and fetch hundreds of pages at once. Fetching hundreds of pages at once is good for the person running the web crawler, but not necessarily good for the person running the web server on the other end. Just imagine how the request queue works in practice. When a web page like www.company.com is fetched, it is parsed and all of the links on that page are added to the request queue. The crawler will then attempt to fetch all of those pages at once. If the web server for www.company.com is not very powerful, it might spend all of its time handling requests from the crawler instead of handling requests from real users. This kind of behaviour from web crawlers tends to make web server administrators very angry. To avoid this problem, web crawlers use politeness policies. Reasonable web crawlers do not fetch more than one page at a time from a particular web server. In addition, web crawlers wait at least a few seconds, and sometimes minutes, between requests to the same web server. This allows web servers to spend the bulk of their time processing real user requests. To support this, the request queue is logically split into a single queue per web server. At any one time, most of these per-server queues are off-limits for crawling, because the crawler has fetched a page from that server recently. The crawler is free to read page requests only from queues that haven't been accessed within the specified politeness window.

When using a politeness window, the request queue must be very large in order to achieve good performance. Suppose a web crawler can fetch 100 pages each second, and that its politeness policy dictates that it cannot fetch more than one page each 30 seconds from a particular web server. The web crawler needs to have URLs from at least 3,000 different web servers in its request queue in order to achieve high throughput. Since many URLs will come from the same servers, the request queue needs to have tens of thousands of URLs in it before a crawler can reach its peak throughput.

```
User-agent: *
Disallow: /private/
Disallow: /confidential/
Disallow: /other/
Allow: /other/public/
User-agent: FavoredCrawler
Disallow:

Sitemap: http://mysite.com/sitemap.xml.gz
```

Even crawling a site slowly will anger some web server administrators who object to any copying of their data. Web server administrators who feel this way can store a file called /robots.txt on their web servers. Figure 4.1 contains an example robots.txt file. The file is split into blocks of commands that start with a User-agent: specification. The User agent line identifies a crawler, or group of crawlers, affected by the following rules. Following this line are Allowed and Disallow rules that dictate which resources the crawler is allowed to access. In the figure, the first block indicates that all crawlers need to ignore resources that begin with /private/, /confidential/, or /other/, except for those that begin with /other/public/.The second block indicates that a crawler named Favoured Crawler gets its own set of rules: it is allowed to copy everything. The final block of the example is an optional Sitemap: directive, which will be discussed later in this section.

Implementation of a crawling threads, using the crawler building blocks we have seen so far. Assume that the frontier has been initialized.

A simple crawling threads implementation

```
Procedure Crawler Thread (frontier)
While not frontier. Done () do
Website ← frontier.nextSite ()
url ← website.nextURL ()
if website.permitsCrawl(url) then
text ← retrieveURL(url)
store Document(url, text)
for each url in parse(text) do
frontier.addURL(url)
end for
```

end if
frontier.releaseSite(website)
end while
end procedure

With a few URLs that act as seeds for the crawl. The crawling thread first retrieves a website from the frontier. The crawler then identifies the next URL in the website's queue. In permits Crawl, the crawler checks to see if the URL is okay to crawl according to the website's robots.txt file. If it can be crawled, the crawler uses retrieve URL to fetch the document contents. This is the most expensive part of the loop, and the crawler thread may block here for many seconds. Once the text has been retrieved, store Document stores the document text in a document database (discussed later in this chapter). The document text is then parsed so that other URLs can be found. These URLs are added to the frontier, which adds them to the appropriate website queues. When all this is finished, the website object is returned to the frontier, which takes care to enforce its politeness policy by not giving the website to another crawler thread until an appropriate amount of time has passed. In a real crawler, the timer would start immediately after the document was retrieved, since parsing and storing the document could take a long time.

4.2.3 Freshness: Web pages are constantly being added, deleted, and modified. To keep an accurate view of the Web, a web crawler must continually revisit pages it has already crawled to see if they have changed in order to maintain the freshness of the document collection. The opposite of a fresh copy is a stale copy, which means a copy that no longer reflects the real content of the web page.

An HTTP HEAD request and server

Client request: HEAD /csinfo/people.html HTTP/1.1
Host: www.cs.umass.edu
HTTP/1.1 200 OK
Date: Thu, 03 Apr 2008 05:17:54 GMT
Server: Apache/2.0.52 (CentOS)
Last-Modified: Fri, 04 Jan 2008 15:28:39 GMT
Server response: ETag: "239c33-2576-2a2837c0"
Accept-Ranges: bytes
Content-Length: 9590
Connection: close
Content-Type: text/html; charset=ISO-8859-1

Under the freshness metric, a page is fresh if the crawl has the most recent copy of a web page, but stale otherwise. Freshness is then the fraction of the crawled pages that are currently fresh. Keeping freshness high seems like exactly what you'd want to do, but optimizing for freshness can have unintended consequences. Suppose that http://www.example.com is a popular website that changes its front page slightly every minute. Unless your crawler continually polls http://www.example.com, you will almost always have a stale copy of that page. Notice that if you want to optimize for freshness, the appropriate strategy is to stop crawling this site completely! If it will never be fresh, it can't help you freshness value. Instead, you should allocate your crawler's resources to pages that change less frequently. Of course, users will revolt if you decide to optimize your crawler for freshness. They will look at http://www.example.com and wonder why your indexed copy is months out of date. Age is a better metric to use.

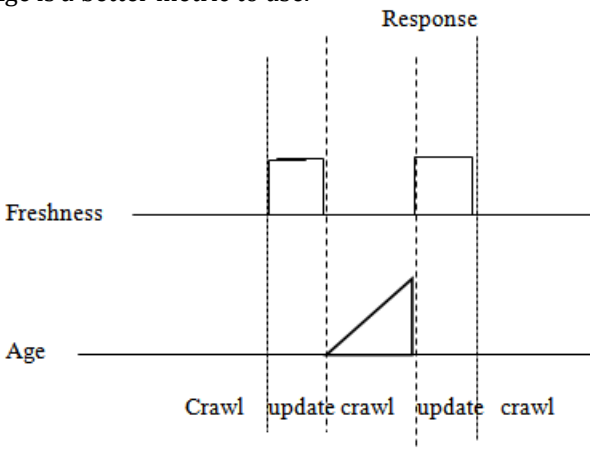

Fig. 4.2 Age and freshness of a single page over time

You can see the difference between age and freshness in Figure 3.6. In the top part of the figure, you can see that pages become fresh immediately when they are crawled, but once the pages changes, the crawled page becomes stale. Under the age metric, the page has age 0 until it is changed, and then its age grows until the page is crawled again.

4.2.4 Focused Crawling: Some users would like a search engine that focuses on a specific topic of information. For instance, at a website about movies, users might want access to a search engine that leads to more information about movies. If built correctly, this type of vertical search can provide higher accuracy than general search because of the lack of extraneous information in the document collection. The computational cost of running a vertical search will also be much less than a full web searches, simply because the collection will be much smaller. The most accurate way to get web pages for this kind of engine would be to crawl a full copy of the Web and then throw out all unrelated pages. This strategy requires a huge amount of disk space and bandwidth, and most of the web pages will be discarded at the end. A less expensive approach is focused, or topical, crawling. A focused crawler attempts to download only those pages that are about a particular topic. Focused crawlers rely on the fact that pages about a topic tend to have links to other pages on the same topic. If this were perfectly true, it would be possible to start a crawl at one on-topic page, then crawl all pages on that topic just by following links from a single root page. In

practice, a number of popular pages for a specific topic are typically used as seeds.

Focused crawlers require some automatic means for determining whether a page is about a particular topic. Once a page is downloaded, the crawler uses the classifier to decide whether the page is on topic. If it is, the page is kept, and links from the page are used to find other related sites. The anchor text in the outgoing links is an important clue of topicality. Also, some pages have more on-topic links than others. As links from a particular web page are visited, the crawler can keep track of the topicality of the downloaded pages and use this to determine whether to download other similar pages. Anchor text data and page link topicality data can be combined together in order to determine which pages should be crawled next

Conclusion and Future Scope

After reviewing the literature of information retrieval in web crawling, it has been concluded that each and every methodology of information retrieval given above supports some unique features or parameters which help in depicting various relationships. On the basis of 21 parameters, the comparative analysis of various strategies of information retrieval has been done. Strategies included in information retrieval for web crawling have been discussed in this paper exhaustively. In a commercial search engine a crawler should have incremental ability to scale seamlessly in a distributed environment with the use of machine learning and focused crawling techniques to provide personalized user experience.

In addition, it is worth to be mentioned here that users' feedback is a precious source of information which is used by many commercial search engines for improving search results.

If such valuable source of information is available, then it is strongly recommended for automatic evaluation methods to utilize it during the process of constructing the relevance judgment set.

We implemented techniques that use various query reformulation strategies in order to help users gain better and more accurate results. Users are provided with a kind of query assistance, being based on the WorldNet. Subsequently, we interconnected the keywords of the pages with the hyperlinking pages of Wikipedia, so as to semantically group the results returned by a search engine, into a hierarchy of labels clusters. The Wikipedia encyclopaedia also facilitates the labelling of the produced clusters.

References

AbdulJaleel, N., & Larkey, L. S. (2003). Statistical transliteration for English Arabic cross language information retrieval. In CIKM '03: *Proceedings of the twelfth international conference on information and knowledge management* (pp. 139–146). ACM.

Agichtein, E., Brill, E., & Dumais, S. (2006). Improving web search ranking by incorporating user behavior information. In SIGIR '06: *Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 19–26).ACM.

Agichtein, E., Brill, E., Dumais, S., & Ragno, R. (2006). Learning user interaction models for predicting web search result preferences. In SIGIR '06*: Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieva*l (pp. 3–10). ACM.

Agichtein, E., Castillo, C., Donato, D., Gionis, A., & Mishne, G. (2008). Finding high-quality content in social media. In WSDM '08: *Proceedings of the international conference on web search and web data mining* (pp. 183–194). ACM.

Allan, J. (1996). Incremental relevance feedback for information filtering. In SIGIR '96: *Proceedings of the 19th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 270–278). ACM.

Balog, K., Azzopardi, L., & de Rijke,M. (2006). Formal modelsfor expert finding in enterprise corpora. In SIGIR '06: *Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 43–50). ACM.

Barroso, L. A., Dean, J., & Hölzle, U. (2003). Web searchfor a planet:The Google cluster architecture*. IEEE Micro,* 23(2), 22–28.

Beeferman, D., & Berger, A. (2000). Agglomerative clustering of a search engine query log*. In Proceedings of the sixth ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 407–416). ACM.

Belew, R. K. (2000). Finding out about. Cambridge, UK: *Cambridge University Press*.

Belkin, N. J. (2008). (Somewhat) grand challenges for information retrieval. *SIGIR Forum,* 42(1), 47–54.

Turpin, A., Tsegay, Y., & Hawking, D. (2007). Fast generation of result snippets in web search. In SIGIR '07*: Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 127–134). ACM.

Vasconcelos, N. (2007). From pixels to semantic spaces: *Advances in conten tbased image retrieval. Computer*, 40(7), 20–26.

Witten, I. H., Moffat, A., & Bell, T. C. (1999). Managing Gigabytes*: Compressing and indexing documents and images San Francisco, CA, USA: Morgan Kaufmann.* Xia, F., Liu, T., Wang, J., Zh, (2nd ed.).

Xu, J., & Croft, W. B. (1998). Corpus-based stemming using cooccurrence of word variants*. ACM Transactions on Information Systems,* 16(1), 61–81.

Yang, S., Zhu, H., Apostoli, A., & Cao, P. (2007). N-gram statistics in English and Chinese: Similarities and differences. In ICSC '07*: International conference on semantic computing IEEE Computer Society.* (pp. 454–460).

Zhang, Y., & Callan, J. (2001). Maximum likelihood estimation for filtering thresholds. In SIGIR '01: *Proceedings of the 24th annual international ACM SIGIR conference on research*.