Design and Implementation of Mobile World Wide Web Search Engines

Varun Krishna, Wen-Chen Hu,
and Aashish Bhatia
Department of Computer Science
University of North Dakota
Grand Forks, ND 58202-9015
varun.krishna@hotmail.com,
wenchen@cs.und.edu,
aashish.bhatia@und.edu

Naima Kaabouch
Department of Electrical Engineering
University of North Dakota
Grand Forks, ND 58202-7165
naimakaabouch@mail.und.edu

Abstract

Mobile users often need to access information on the World Wide Web anytime or anywhere. A common approach is to use Internet-enabled mobile handheld devices such as smart cellular phones to search the Internet. However, finding the appropriate queries for Web search engines is never an easy task and the search results may or may not be relevant to what the users are really looking for, as the information displayed on mobile phone is highly volatile, distributed, and hetrogeneous. Also, there are constraints inherent in mobile handheld devices, such as slow communication, low storage capacity, and awkward input methods. Due to the factors, information discovery by using handheld devices becomes impractical and inconvenient. This research designs and implements a mobile World Wide Web search engine, which includes three major components:

- *Crawlers*, which are used to collect mobile Web pages by using a breadth-first search method.
- *Indexing software*, which indexes the Web pages collected by crawlers for fast searching and retrievals. A database will be used to store and search the indexing information.
- *Searching and ranking software*, which is used to retrieve and rank the search results.

Related research such as the mobile Web and handheld devices will also be discussed in this paper.

1 Introduction

One of the most common tasks performed on the Web is to search Web pages, which is also one of the most frustrating and problematic. The situation is getting worse because of the Web's fast growing size and lack of structural style, as well as the inadequacy of existing Web search engine technologies (Lawrence & Giles, 1999). Traditional search techniques are based on users typing in search keywords which the search services can then locate the desired Web pages. However, this approach normally retrieves too many documents, of which only a small fraction are relevant to the user's need. Furthermore, the most relevant documents do not necessarily appear at the top of the query output list. A number of corporations and research organizations are taking a variety of approaches to try to solve these problems. These approaches are diverse and none of them dominates the field. The approach used in this research explores the possibility of improving the search results according to the user's requirement.

This paper presents an approach to speed up the information retrieval in mobile handheld devices. To achieve this, autonomous crawlers are sent to the Internet for collecting data from hypertext links on that Web page and discover other pages related with the current Web page. The crawlers after extracting information from the Web pages store the information in database. A user can enter the keyword in a search engine interface, which then executes a query and returns the result sets from the database. The search engine is composed of three components:

- *Crawlers,* which is a program that automatically scans various Web sites and collects Web documents from them. Crawlers follow the links on a site to find other relevant pages. Two search algorithms, breadth-first searches and depth-first searches, are widely used by crawlers to traverse the Web.
- *Indexing software*, which is the process of algorithmically examining information items to build a data structure that can be quickly searched. Filtering (Baeza-Yates, 1992) is one of the most important pre-processes for indexing. Filtering is a typical transformation in information retrieval and is often used to reduce the size of a document and/or standardize it to simplify searching.
- *Searching and ranking software,* which is used to retrieve and rank the search results. Query processing is the activity of analyzing a query and comparing it to indexes to find relevant items. A user enters a keyword or keywords, along with Boolean modifiers such as "and," "or," or "not," into a search engine, which then scans indexed Web pages for the keywords. To determine in which order to display pages to the user, the engine uses an algorithm to rank pages that contain the keywords (Zhang & Dong, 2000).

This paper work introduces a new focused search methodology for pervasive devices satisfying the constraints dictated by those devices and also their operational behavior mode.

2 Background Studies

A search engine as shown in Figure 1 usually collects Web pages on the Internet through a spider also known as crawler or robot software, all of which will be scanned and indexed based on the full text of documents (Hu, Yang, Yeh, & Lee, 2004). In a typical search procedure, the user submits a query, which is simply a word or combination of words as keywords. The search engine will examine its backend database for any document found in the index which matches

the query, and then return a list of related Web pages. In this way, a Web user can quickly obtain a set of all the Web pages in the search engine's database containing the given keywords.

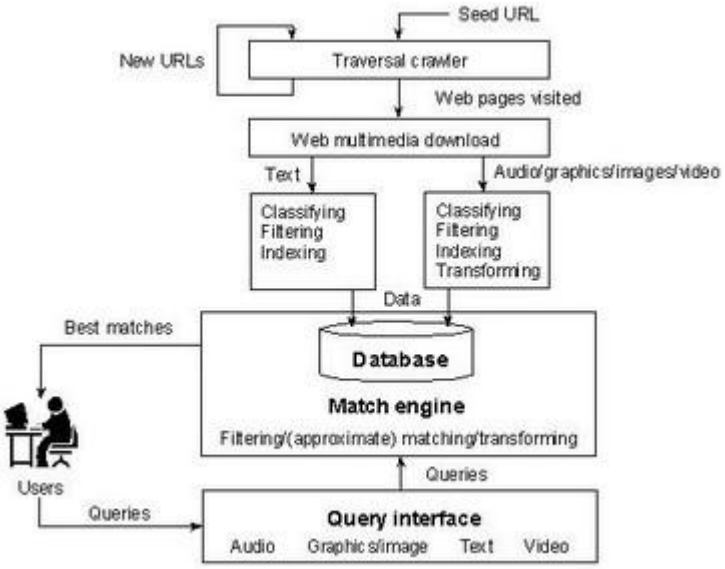


Figure 1: A Generic System Structure of Search Engines.

2.1 Retrieval Algorithms

Two kinds of retrieval algorithms are available (Raghavan, 1997):

- *Sequential scanning of the text*: Sequential scanning involves reading the text line by line to search for a pattern or information. In this thesis, sequential scanning of source code is performed. The file content is considered to be a string of text and is scanned to find a pattern using regular expressions. Sequential scanning does not need extra memory. The running time is proportional to the size of the text.
- *Indexed Text*: Indexing involves preparing a set of documents or URLs for searching. A simple look-up in the indexed URLs retrieves all search results which contains at least one of the search terms. In this thesis the indexed URLs are stored in the database. This speeds up the search. The index size is proportional to the database size.

2.2 Query Processing

Query processing is the activity of analyzing a query and comparing it to indexes to find relevant terms. A user enters a keyword or keywords, along with Boolean operators such as "and,", "or,", and "not," into a search engine, which then scans indexed Web pages for the keywords. In this thesis, the index text approach is followed for mobile information retrieval. Here, index terms are stored in the database for each of the information content. When the user enters the query, the keywords are compared with that of the index terms associated with documentary information stored in the database and the results are displayed on the screen.

2.3 Filtering Algorithm

Filtering is one other aspect, which is very essential in mobile information retrieval. Filtering is a typical transformation in information retrieval. It is a method to simplify searching in general and to provide relevant output. Major filtering techniques used in this research include:

- Common words are removed using a list of stop words such as "of" and "the", which make poor index terms. Stop word list may be entered into a search statement but cannot be searched for as individual words.
- Special symbols such as '@' removed and sequences of multiple spaces reduced to one space.
- All the URL's which are linked with a pdf (Portable Document Format) file are not inserted into the database.

All these features considerably reduces the number of irrelevant results.

2.4 Indexing

Almost all types of indexes are based on some kind of trees or hashing, except clustered data structures, and direct acyclic word graph. Indexing is generally a process of assigning some information or keywords for the documents so that the entire search process can be done quickly. The query terms entered by the user are compared with the indexed terms and the close results are displayed. There are many techniques employed by the search engines for indexing. Some of them include:

- *Content*: Page content provides the most accurate and full-text information. However, it is least used as context extraction is far less practical.
- *Descriptions*: Page descriptions can either be constructed from the metatags or submitted by Webmasters or reviewers.
- *Hyperlinks*: Hyperlinks contain high-quality semantic clues to a page's topic. A hyperlink to a page represents an implicit endorsement of the page pointed to (Chakrabarti et al., 1999).
- *Hyperlink text*: Hyperlink text is normally a title or brief summary of the target page.
- *The first sentence*: The first sentence of a document is also likely to give crucial information related to the document.

3 The Proposed Mobile World Wide Web Search System

The amount of Web pages is growing rapidly, as well as the number of users in Web search. The rate of the Web page growth has been and continues to exponential. In 1994, one of the first Web search engines, the World Wide Web Worm (WWWW) had an index of 110,000 Web pages and Web accessible documents. Antonio and Alessio (Antonio & Alessio, n.d.) estimated there was 11.5 billion Web pages as of the end of January 2005 and Nathan Weinberg (n.d.) count 25 billion Web pages Google has indexed by the time January 2006. One can guess at this rate the most popular site Google needs to index roughly 400 billion Web pages in 2010. Creating a search engine which deal with increasing Web growth presents many challenges. Fast crawling technology is needed to gather the Web pages and keep them up to date. Storage space must be used efficiently to store indices. The indexing system must process tens of petabytes of data efficiently and queries must be handled quickly. These tasks are becoming increasingly difficult

as the Web grows. A mobile World Wide Web search system is proposed in this research. This section gives a detailed description of the system which includes three major components: (i) crawling, (ii) indexing, and (iii) searching and ranking software.

3.1 The System Structure

This section explains the different elements/tiers used in the proposed system. In this research, a model of search engine is implemented which consists of three tiers namely the presentation tier, logic tier and database tier.

- The presentation tier is the user interface which is seen by the users of the system. All the indexing, display of results and deletion of the indexed Web pages are displayed in this tier and hence this tier is at the top level of the application.
- The logic tier is the second tier which has the application logic to retrieve the search results from the search engine which is implemented, discovering the related page elements that need to be displayed in the mobile browser, ranking the elements etc. This tier communicates with the presentation tier to display the results after the logic is being applied. It also communicates with the database tier which is the third tier used in this research to store or retrieve the related data needed by the application.
- The database tier is like a repository which is used to store the required data/information needed by the application. This data can be retrieved, modified or refined and saved back to the repository for the application use. The repository/database used in this research is an object - relational database management system, Oracle 9i. A generic three tier architecture is shown in Figure 2.

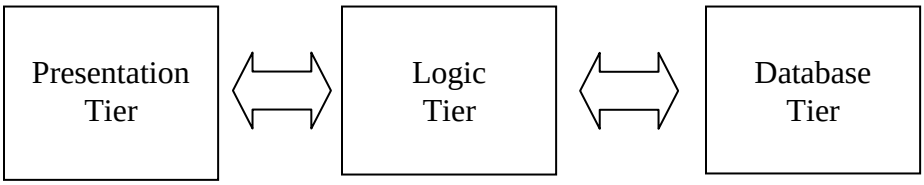


Figure 2: A Generic Three Tier Architecture.

The logic tier used in this research consists of the search engine which retrieves the results from indexed Web pages which are stored in the database. The logic tier refines the results and then is being sent to the presentation layer to display as results. Figure 3 gives an overview of this research with the above mentioned tiers. The logic tier has the search engine and also the information retrieval system. The information retrieval system is used to find the related page elements based on the search. These elements are assigned a rank based on their relevance to the search keyword. The information retrieval system retrieves only those elements that are necessary, eliminating the unnecessary information/elements.

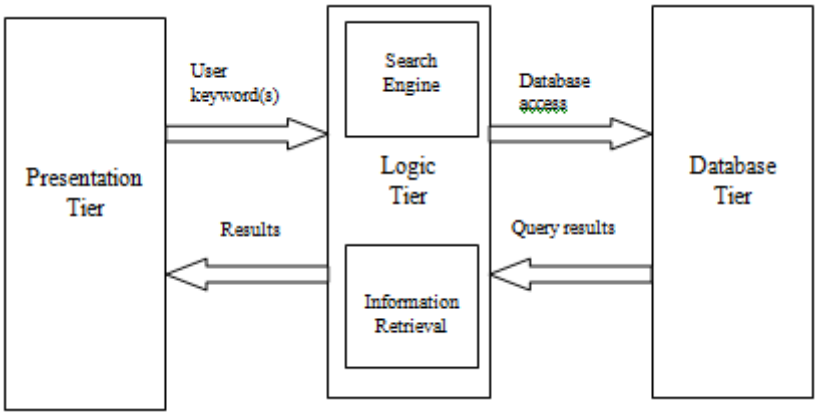


Figure 3: Three Tier Architecture Used in this Research.

The proposed system includes database at the server and interfaces on handheld devices for browsing. A search engine is developed for searching the data. It consists of a crawler, indexing, searching and ranking software:

- The crawler is mainly developed to search the Internet, more specifically the mobile Internet. A seed URL is provided to the crawler, which uses it to explore the Internet for Web-page collection.
- Indexing can be described in general as the act of associating keywords as pointers to the set of documents. They are a group of terms or phrases. In this system, the hyperlink text information collected by the crawler is used to index the URLs.
- Searching and ranking are used to retrieve and display the results.

Finally, software is written for information retrieval from the database. The following subsections discuss its components and their implementation.

3.2 Crawling

The crawler is also called robots or spider; they are the software programs that automatically scan the World Wide Web by retrieving URLs, keywords, links, text, etc. Crawlers follow the links on a site to find other relevant pages. Crawlers start by parsing a specified Web page, noting any hypertext links on that page that point to other Web pages. They then parse those pages for new links recursively. Crawler software are not viruses which move around to different computers on Internet but reside on a single machine and send HTTP requests for documents to other machines on the Internet, just as Web browser does when the user clicks on the links.

3.2.1 Crawler Components

The crawler/spider program is written in Java and as the program is called, it downloads the content of a seed URL and then scans and stores all the URLs links of the specified Web page. Several ways can be used to discover the URL of an HTML file. In my program the spider collects all the URLs by picking out the string following HTML tag "<a href=". A queue of "Array" data type is used to store the URLs. Initially, it consists of seed URLs. If the queue is

terminated, i.e., if there are no more URLs present, the crawler stops. The crawler takes the URLs one at a time, and explores the Web to retrieve the content of selected URL. The retrieved Web content is parsed, i.e., the text is processed and the important information is collected and stored. The URLs present in this Web content are added at the end of the queue and the process is repeated for each URL. Few functions are written for the crawler implementation. The "lynx" system call is used to download all the Web content. It is a powerful text browser for the World Wide Web. It displays HTML documents containing links to files residing on the local system. The crawler process basically consists of five sub processes which help to retrieve and store information. The details of the processes are as follows:

1. *Constructor*: The first method is a class "Constructor" which calls all other methods. The function that is called first from this constructor is "Start Process," which, in turn, calls the methods "Lynx Implementation" and "Text Process." These methods dump the Web page content of the seed URL, parse it and store it in the corresponding arrays. Figure 4 shows an example of a seed URL, which is used by the crawler as a starting point to explore the Web.

Figure 4: Seed URL Input.

   Once the hyperlinks from the seed URL are stored, the method, "Start Process," is called repeatedly. URLs stored in the queue are fetched one by one continuously in first-in-first-out manner by this method. The selected URL is again processed by the "Lynx Implementation" and "Text Process" functions. The URLs and hyperlink texts retrieved from this Web page are added at the end of respective arrays. The entire process is repeated for each URL. A breadth first approach is followed in carrying out the above process.
2. *Start Process*: This method calls "Lynx Implementation" and "Text Process" functions. The first method dumps the Web content and the latter parses it. These functions are called repeatedly for each selected URL from the queue array. The entire process is repeated until all the URLs present in the queue are explored or there is no more information to be crawled.
3. *Lynx Implementation*: This method uses a "lynx" system call to retrieve the Web page content of the seed URL at the start of the crawling process. The following shows how to use the "lynx" system call:

```
/usr/bin/lynx -dump   seedURL
```

   The content is usually stored in a buffer, or sometimes files, to be used in the "`Text Process`" method. The same function is used when successive URLs taken from the queue are processed.

4. *Text Process*: The Web content stored in the buffer or file is parsed by this method for hyperlink text and URLs. These URLs and hyperlink text are added at the end of their respective arrays which are used in the Information Storage method. All the above functions generally perform the crawler functionality, and the last method is basically used to index the URLs and store them in the database.
5. *Information Storage*: This method collects the hyperlink text and URLs from the respective arrays and stores them in a database. While storing data in the database, the URLs are indexed with the hyperlink text. This hyperlink text is mainly used as keywords while performing a search as it describes the information or contents of the Website to a maximum extent. These are the functions written in implementation of the crawler for indexing the information and storing them in the database. Due to the limited resources available, only a small number of URLs are stored to show the functionality of the system. In general, a large repository is maintained to store the information.

3.3 Indexing

The indexing mainly deals with the extraction of the meta-data including title, keywords and description out of the HTML source file. Since the meta-data only exists in the head section, thus we only need to scan the very first part of the HTML file of about 40-50 lines. After fetching out the closely related term or text from each Web page, all these words are checked in stopwords file and then all the stopwords for example "a," "of," "and," "the," etc. are removed. After removing the stopwords, the remaining words in an array are termed as keywords and are stored in the database. These keywords in the database are now individually parsed into each URL and the number of hits for each stored URL is identified and stored again in the database. Each keyword is taken from the array and is compared with the URLs. The words or phrase with a good match are used as index terms for that URL. The searching takes place with the help of matching these keywords with the user input. Figure 5 shows an algorithm for a piece of code written to store indices and URLs in the database.

```
ALGORITHM Information Storage {
1. Implement the SQL queries
   1.1 query1 = "insert into table(keyword,urls,hits) values(?,?,?)";
   1.2 PreparedStatement ps = conn.prepareStatement(query1);
2. While (hyperlink text titles array and URLs array not finished)
   {
     2.1 Take one hyperlink at a time and compare with the URLs
         generated by the query.
     2.2 Check for the relevance of the hyperlink.
     2.3 Retrieve the corresponding URL.
     2.4 Store the hyperlink text as index and the corresponding URL
         in the database.
   }
} //End
```

Figure 5: Algorithm for Storing Information.

3.4 Searching and Ranking Software

When a user submits a query, the search engine will go through the indexes to find the relevant Web pages and displays the search results on the screen. The order of the displayed results is based on a ranking methodology.

3.4.1 Searching

Searching is a process of finding the required information from the database. Here, a typical search technique is followed by the local small scale search engine, which is simple and straight forward. Since the query design is not the focus of this research, all query designs are straight forward. In the first phase, the main task is to store the result URLs selected by the ranking algorithm into the database, which is done by a typical insert statement. In the second phase, for the purpose of checking downloads, given a query term, it will report all the records with the same query string. Therefore, the query is a select statement. In the last phase, a select statement is constructed to look up an approximate query string match in the field of the record that user specifies. Figure 6 shows a piece of algorithm from the above discussion.

```
ALGORITHM FOR SEARCHING {
 1. Check whether the entire string matches the indexes.
    If yes, store the result.
 2. Split the string entered by the user.
 3. Check for the presence of individual words with the query given by
    3.1 query = "select distinct keyword, url, title, hit from urltable where
    lower(title) like lower ('%"+input[i]+"%')"; where i ranges.
 4. While (resultset not finished) {
      4.1 Retrieve the index and corresponding URL.
    }
 5. Return the search results.
} // End
```

Figure 6: Algorithm for Searching Information.

3.4.2 Ranking

Ranking is a methodology generally used in displaying the search results. The order of results displayed in most of the search engines is decided by following the ranking methodology. In this system, a simple ranking methodology is proposed, where search term frequency is used as a primary way of determining whether a document is relevant. If you're researching diabetes and the word "diabetes" appears multiple times in a Web document, it's reasonable to assume that the document will contain useful information. Therefore, a document that repeats the word "diabetes" over and over is likely to turn up near the top of your list. Figure 7 shows the information retrieval architecture used in this system. The user enters the query which is passed to a program. The program calls the JDBC program which establishes a connection with the database. Proper results are fetched from the database and are then displayed on the screen.

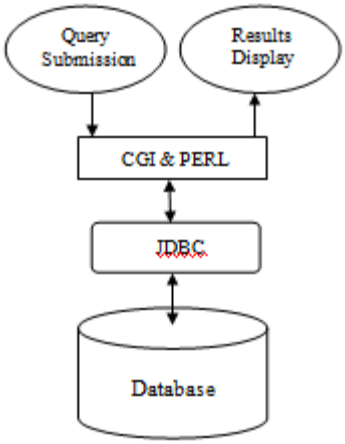


Figure 7: A System Structure of Database-Driven Web Sites.

4 Experimental Results

This section gives a few scenarios demonstrating various information discoveries. There are, however, some factors that need to be considered in implementing the entire system. First is the number of URLs stored in the database. Due to the limited capacity of the Oracle student account, only few URLs are stored, though a large number of URLs are generated.

4.1 Experiment Set-up

To confirm the effectiveness of the proposed method, experiments are conducted to evaluate the indexing, crawling, searching, ranking, usability, and the performance on mobile phones. There are several mobile Web browsers available for testing and implementation of mobile applications. One of them is "Opera Mini simulator" hosted at http://www.operaMini.com.

4.1.1 Opera-Mini Simulator

Opera Mini (Opera Software ASA, n.d.a) as shown in Figure 8 is a Web browser designed primarily for mobile phones, smart phones and PDAs. It uses the Java Micro Edition (ME) platform and consequently requires that the mobile device be capable of running Java ME applications. Opera Mini is offered free of charge, supported through a partnership between its developer, the Opera software company, and Google (Goldman, 2008). Opera Mini was derived from the Opera Web browser for personal computers, which has been publicly available since 1996. Opera Mini began as a pilot project in 2005. It requests Web pages through the opera software company's servers, which process and compress them before relaying the pages back to the mobile phone. This compression process makes transfer time about two to three times faster, and the pre-processing smoothes compatibility with Web pages not designed for mobile phones. Unlike ordinary Web browsers, Opera Mini fetches all content through a proxy server that reformats Web pages into a format more suitable for small screens (Sony Ericsson, 2008). A page is compressed and then delivered to the phone in a markup language called OBML (Opera Binary Markup Language).

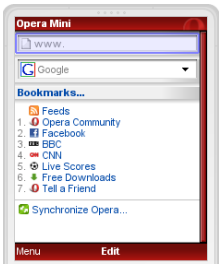


Figure 8: A Screenshot of Opera Mini Simulator.

4.1.2 Other Tools and Software Used in this Research

The proposed system uses Oracle 9i as the database at the back end of the system, and all the programs are hosted with the shell server of the computer science department. JDBC is used to connect the Oracle 9i database server and the Spider program which is also made in Java. JDBC is JavaSoft's database connectivity specification. It is a Java API that enables Java programs to execute SQL statements. JDBC makes it possible to write a single database application that can run on different platforms and interact with different DBMSs. This allows Java programs to interact with any SQL-compliant database. Scripting language Perl is used to make the segmentation software which in turn is linked with CGI and HTML. All the interfaces are browsed by using Opera Mini simulator. The system details are listed in Table 1.

Software	Model/Version/Type
Server	GNU/Linux 5.3 at gandalf.aero.und.edu
Web Server	Apache/2.2.3 (Red Hat) Server
Database server	Oracle 9i
Programming languages	Perl, CGI, HTML,SQL, JDBC, and Java
Text browser	Lynx
Microbrowser	OperaMini::http://www.operaMini.com/demo

Table 1: System Information of this Project.

4.2 Experiments

To confirm the effectiveness of the proposed method in this research, experiments have been conducted to evaluate the segmentation accuracy, its usability, and the performance on mobile phones. Details of the experimental conditions, accuracy evaluation, and usability experiments are presented in the coming sections. One of the important things of any handheld device is the interface usability. The interfaces must make the browsing and navigation easy for users. Considering these points, few interfaces are designed to make the entire system feasible and easy to use. Initially, we have three options to choose from:

- *Index Web Page*, which starts the crawling process of gathering information from the seed URL and implements the database,
- *Search Web Page*, which is used to search for the URL according to the keyword match, and
- *Clear System*, which is used to clear the existing database entry.

To start with the experiment a database is created by indexing the seed URL by choosing the "Index Web Page" option which allows user to create their database for any Website of their interest. Due to limited capacity of the Oracle student account as stated earlier, only few URLs are stored, though a large number of URLs can be generated. Each database containing URL is stored in the backend server. After the database is created, the users can perform a search by submitting a query. The experiment implementation will be discussed in detail and will be explained along with the interfaces.

Figure 9: The System Entry Page.

Figure 9 shows the system entry page. Users can select any one of the options from the list "Index Web Page," "Search Web Page," and "Clear System" and begin the search on the mobile World Wide Web. When the first option is selected, a second interface screen is called which asks for user input of maximum pages to be indexed and a seed URL. After filling in the information, the crawler is called. The crawler explores the Web and retrieves all the required information and stores in the database server. Opera contains a feature known as mobile view, which shows the results in a more compressed and concise form. The mobile view also increases the visibility clearly and decreases the font size automatically to fit in, inside the small mobile screen.

Figure 10: The Interface for Indexing.

4.2.1 Crawling Demonstration

Figure 10 displays demonstration fed by the user. If the user selects "Index" and starts it, a database for this topic is created. Once indexing is done, the crawler, which goes from one Web page to several numerous Web pages, stores the specific information from each Web page. Information like title, keyword, and URL are stored in the database. This is done by the crawler on selection of a demo. There is also a "Home" option which takes the user back to the main interface. The screenshot below shows the confirmation message stating that Indexing has been done successfully.

Figure 11: Query Entry Page after Indexing.

4.2.2 Query Formulation

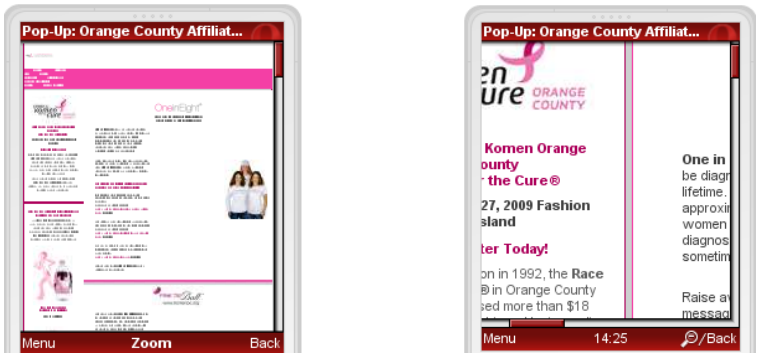
One of the important features in the interface of Figure 11 is the "Search." On selection of this, the user is taken to the interface where the actual process of querying, query formulating, and searching takes place. This interface forms the basis for information retrieval. Figure 12 shows the query formulation interface from where users can enter their keywords for formulating queries. It shows the search screen (on the left) and results after searching the database after entering the keyword "survivor" (on the right). Figure 12.b shows the search results after using the "Search" button. All the results are sorted according to the basic ranking algorithm, with highest ranking at the top of the result page. Thus, refining the query reduces the navigation time and also increases the relevance of the results.

(a) (b)

Figure 12: (a) The Search Keyword and (b) Results from Searching the Database.

4.2.3 Browsing the Search Results on Mobile Browsers

It is observed that when the original Website is viewed on Opera Mini Web browser, it gives tremendous problems for browsing the content. The large amount of content available on the Website, makes it hard for the user to locate the specific information that he/she is looking for, which in turn causes difficulty in navigating through the page. A feature of Opera Mini compresses the images and content and scales them down to more than 80% of the screen size in either direction. Figure 13 shows one of the search results browsed on mobile phones. Since the initial display is unreadable, Opera provides a zoom-in feature (Opera Software ASA. , n.d.b.). It is observed that when the user opens the Web page, the zooming feature is required for viewing the content of the Web page and when the user zooms in the Web page then he/she gets completely lost while navigating the content he/she is looking for.



(a) (b)

Figure 13: (a) One of the Search Results, http://www.oneineight.org/, on Opera Mini and (b) Using the Zoom -in -out Feature of Opera Mini.

5 Conclusion

This research proposes an approach for mobile information retrieval and display. It uses a focused search. According to this approach, efficient browsing of the World Wide Web on mobile phone is possible in an effective manner. Also, the usability of mobile search interfaces is improved by using this system. A user on the move can search and browse the Web easily and fast. By using the proposed system, retrieved information can be conveniently accessed from a handheld device as it is accessed from a desktop. A simple search engine is developed in this research. Though it cannot be compared with major commercial search engines, an effort has been made to achieve the basic search engine functionality to a maximum extent. Autonomous crawlers, simple indexing, and searching and ranking software are also implemented to achieve this purpose.

Traditional text based search engines, which rely on keyword matching, visit World Wide Web sites, fetch pages and analyze text information to build indices, but with the explosive growth in the amount of Internet information, the number of documents in the indices has been increasing by many order of magnitude. One of the advantages of using this system is search results and

speed are greatly improved compared to other search engines. Disadvantages of using this system include (i) this research does not include advanced Boolean features of the search engines and (ii) it is a focused search engine instead of a generic one. The future work may include the following:

- Add simple features which are supported by commercial search engines like Boolean operators, negation, and stemming. Stemming means a linguistic analysis to get to the root form of a word. For example, if the user enters viewer as the query, the search engine reduces the word to its root (view) and returns all documents containing the root.
- Include relevance feedback, which is a feature of information retrieval systems. The idea behind relevance feedback is to take the results that are initially returned from a given query and to use information about whether or not those results are relevant to perform a new query.
- Include suggestion(s) if the query fails in accordance with the database.
- Include clustering, which is a technique the search engines use to group different pages from the same domain in their search results pages.

References

Antonio, & Alessio. (n.d.). Search Engine Watch. Retrieved December 09, 2009 from http://www.searchenginewatch.com

Baeza-Yates, R. A. (1992). Introduction to data structures and algorithms related to information retrieval. In W. B. Frakes and R. A. Baeza-Yates, editors, *Information Retrieval Data Structures & Algorithms*, pages 13-27, Prentice-Hall.

Chakrabarti, S., Dom, B. E., Kumar, S. R., Raghavan, P., Rajagopalan, S., Tomkins, A., Gibson, D., & Kleinberg, J. (1999). Mining the Web.s link structure. *IEEE Computer*, 32(8):60-67.

Goldman, D. (2008). Back to Google on Opera mini and Opera mobile. *Opera Watch*. Retrieved December 05 2009, from http://operawatch.com/news/2008/02/back-to-google-on-opera-mini-and-opera-mobile.html

Hu, W.-C., Yang, H.-J., Yeh, J.-h., & Lee, C.-w. (2004). World Wide Web search technologies. In Mehdi Khosrow-Pour, editor, *Encyclopedia of Information Science and Technology*, Vol. I-V, pages 3111-3117, IRM Press.

Lawrence, S. & Giles, C. L. (1999). Accessibility of information on the Web. *Nature*, 400:107-109.

Opera Software ASA. (n.d.b.). *Opera's Small-Screen Rendering*. Retrieved June 23, 2009, from http://www.opera.com/products/mobile/smallscreen/

Opera Software ASA. (n.d.a.). *Opera Mini – Features*. Retrieved December 29, 2009, from http://www.opera.com/mini/features/

Raghavan, P. (1997). Information retrieval algorithms, a survey. *LATIN 2000*: 123-125

Sony Ericsson. (2008). *Developer Case Study: Managing Java Fragmentation, Opera Software's Java ME Browser Client: About Opera Mini*. Retrieved December 29, 2009, from http://developer.sonyericsson.com/site/global/newsandevents/latestnews/newsjune06/p_opera_mini_java_casestudy.jsp

Weinberg, N. (n.d.). *Search Engine Showdown*. Retrieved November 07, 2009, from http://searchengineshowdown.com

Zhang, D. & Dong, Y. (2000). An efficient algorithm to rank Web resources. In *Proceedings of the 9th International World Wide Web Conference*, Amsterdam, Netherlands.