AIN SHAMS UNIVERSITY
Faculty of Computer &
Information Sciences
Scientific Computing Department

An Efficient Ranking Module for an Arabic Search Engine

Thesis submitted to the Department of Scientific Computing
Faculty of Computer and Information Sciences
Ain Shams University

In partial fulfillment of the requirements for the degree
of Master in Computer and Information Sciences

By

Esraa Abd El-Raouf Hamed Abd El-Hady
B.Sc. in Computer and Information Sciences
Ain Shams University – Cairo

Under the supervision of

Prof. Dr. Mohamed F. Tolba
Basic Science Department
Faculty of Computer and Information Sciences
Ain Shams University

Dr. Nagwa Badr
Information Systems Department
Faculty of Computer and Information Sciences
Ain Shams University

Dr. Mohamed Abdeen
Computer Science Department
Faculty of Computer and Information Sciences
Ain Shams University

Cairo-2011

Acknowledgement

I acknowledge my deep gratitude to ALLAH the most beneficent and most merciful, who helped me complete this work on a level that I hope will please the reader.

First of all I would like to thank my dear family, my husband Geo. Mahmoud Eissa, Mother, Father and brothers whose kindness, care and never ending support and encouragement made me the person I am today.

Special thanks are due to my supervisors; Prof. Dr. Mohamed Fahmy Tolba, Dr. Nagwa Badr, and Dr. Mohamed Abdeen at the Faculty of Computer and Information Sciences, Ain Shams University, who constantly guided me in elaborating this thesis, assisted me in understanding and analyzing problems, and continuously provided support and valuable comments.

I would also like to thank Dr. Hossam Mahgoub who supports me with his morphological meanings database.

Finally I would like also to thank all my friends who pushed me either directly or indirectly to finish this work in the moments I thought it never will.

Abstract

The plentiful content of the World Wide Web is useful to millions. Some simply browse the Web through entry points. But many information seekers use a search engine to begin their Web activity. In this case, users submit a query, typically a list of keywords, and receive a list of Web pages that may be relevant, typically pages that contain the keywords. Now, search engines became very essential information resources for net users and they form a very important commercial industry.

Searching online provides you with a wealth of information, but not all of it will be useful or of the highest quality. Search engines are distributed programs that dive into the World Wide Web to find relevant information for a given search query. Their fundamental components are: the crawlers, the indexer module, the collection analysis module, the query engine, and the ranking module.

The ranking module represents a significant component in web search engines. The main function of the ranking module is to sort the search results by relevance or importance using information retrieval (IR) algorithms.

There were two kinds of methods in information retrieval, based on content and based on hyper-link. The quantity of computation in systems based on content was very large and the precision in systems based on hyper-link only was not ideal. It was necessary to develop a technique combining the advantages of two systems.

Many web users are interested in Arabic web browsing whether the reason is academic or commercial… suffer to find their search and request over the Arabic search engine etc. As the existing web search engines are designed to perform English web searches. They don’t generate morphological variations of Arabic words but they just match the word as it is. Therefore their results contain only the pages that exactly match the user query. They also don’t consider the different meanings of a word so search results contain unrelated pages to user query.

In this research, we focus on implementing an enhanced ranking algorithm by combining both the page content and the Hyper-Link with the focus on Arabic search engines by taking into account the stem and the context of the Arabic word by combining both the count of words related to query in the page and the count of words related to query in outlinks pages of that page to calculate its rank, using external database having the morphological meanings of the most Arabic words. Then sort the pages according to its rank.

If there is more than one meaning to an input query word in case the user does a query in using only one word, the user may choose the meaning he/she wishes to search for. The search results will largely contain the inflected forms of the word that belong to that meaning. This helps reduce the redundancy that results from morphological search only.

Distributed page ranking are needed because the size of the web grows at a remarkable speed and centralized page ranking is not scalable.
To speed up the ranking module process, this thesis proposes a parallel technique for this Arabic ranking module. We applied this Arabic ranking module on a dataset of 10000 Arabic web pages. This research proved that the optimal number of processors needed for this parallelization is 10 processors. Using this number of processors, the proposed parallel algorithm is very efficient and gets perfect speedup.

Table of Contents

Page

Acknowledgement

Abstract…………………………………………………………………... I

Table of Contents………………………………………………………... III

List of Figures…………………………………………………………. VI

List of Tables…………………………………………………………. VII

1- Introduction **1**

1.1 Preliminaries………………………………………………. 3

1.1.1 The Crawling Module…………………………………….. 3

1.1.2 The Indexing Module…………………………………….. 4

1.1.3 The Page Repository……………………………………… 6

1.1.4 The Query Engine………………………………………… 7

1.1.5 The Page Ranking Module……………………………….. 8

1.2 Ranking the Web…..…………….……………………………. 9

1.3 Motivation……………………………………………………… 11

1.4 Objectives……………………………………………........ 12

1.5 Thesis Organization……………………………………….. 13

2- A Survey on Web Ranking **14**

2.1 Introduction ………………………………………………………. 14

2.2 Factors Contributing to Ranking …………………………………….. 16

2.2.1 Domain name ………..…………………………………….. 16

2.2.2 Website URLs ……………………………………………….. 18

2.2.3 "On site" or "On page" SEO parameters - Header info..…. 20

2.2.4 "Offsite" or "Off page" parameters - Link popularity …… 25

2.3 Related Work……..……………………………………………. 28

2.4 Why using morphological analyzer……………………………… 37

2.5 Parallel Ranking……………………………………………….. 38

2.6 Conclusion………..……………………………………………. 39

3- The Architecture of our Arabic Web Ranking Technique and its parallelization 41

3.1 The Architecture of our Arabic Web Ranking Technique …...… 41

3.1.1 Our Proposed Architecture …………..………………… 41

3.1.1.1 Ranking Module (A)…..……………..…………..… 43

3.1.1.2 Ranking Module (B)……………..…………..… 45

3.2 The Architecture of Parallelization for our Ranking technique…. 48

3.2.1 Our Proposed Parallel Architecture ……………………. 50

3.2.1.1 Steps of parallelization…………………………. 50

4- The Implementation of our Arabic Web Ranking Technique and its Parallelization 52

4.1 The Implementation of our Arabic Web Ranking Technique….…. 53

4.1.1 The Experiment ………………………………………… 54

4.1.1.1 Pre-process steps………………………………. 55

4.2 The Implementation of Parallelization for our Ranking technique... 57

4.2.1 The Experiment ………………………………………… 59

5- The Evaluation of our Arabic Web Ranking Technique and its Parallelization 61

5.1 Performance Results for the Proposed Arabic Ranking Technique 61

5.1.1 Yahoo search engine results……………………………. 63

5.1.2 Google search engine results……………………………. 67

5.2 Performance Results for a Parallelization based technique for a High Performance Arabic Ranking…………………………….. 69

5- Conclusions & Future Work **74**

6.1 Summary………………………………………………………………. 74

6.1.1 An Efficient Arabic Web Ranking ……….……………… 75

6.1.2 Efficient Parallelization for our Ranking Technique…...… 76

6.2 Future Work……………………………………………………….. 78

6.2.1 Factors Contributing to Ranking………………………….. 78

5.2.2 Deal with more than one Arabic word …………..………. 78

References…………………………………………………………..... 79

Publications

Arabic Summary

List of Figures

Fig. 1.1 Hyperlink Structure of a 6-node sample Web.…………… 1
Fig. 1.2 The general architecture of search engines…….............. 2
Fig. 1.3 The basic stages of the indexer…………….…………….. 5
Fig. 2.1 Factors contributing to Rankings………………………… 15
Fig. 2.2 The reverse tree Architecture……………………….……. 18
Fig. 2.3 Websites links…………………………………….……… 26
Fig. 2.4 An Authority node and a hub node…………….………… 29
Fig. 2.5 PageRank……………………………………………….. 30
Fig. 3.1 Our proposed Architecture ……………………………... 42
Fig. 3.2 Steps of Ranking Module (A)……………………………. 44
Fig. 3.3 Steps of Ranking Module (B)....……………………..……... 46
Fig. 3.4 Our proposed Parallel Architecture……………………… 50
Fig. 4.1 The steps of our prototype……………………………….. 53
Fig. 5.1 Yahoo search engine results and our proposed method results compared with the five interested user…………….. 66
Fig. 5.2 Google search engine results and our proposed method results compared with the five interested users…………… 68
Fig. 5.3 Elapsed time versus number of processors……………... 71
Fig. 5.4 Speedup versus number of processors.……………….…. 72
Fig. 5.5 Efficiency versus number of processors…………………… 73

List of Tables

Table 5.1	Yahoo search engine results compared with our propose method results according to five interested users................	62
Table 5.2	Google search engine results compared with our proposed method results according to five interested users………..	67
Table 5.3	Table of time elapsed in Ranking Module (A) with no of processors…………………………………………………	70

Chapter 1

Introduction

Search engines are enabling tools to mine the massive information content of the World Wide Web. They are very useful for many information seekers on the web. They have evolved over the last two decades until reaching the level of maturity we see today, encompassing, quick, and easy to use.

There are many challenges in building good search engines. One of these challenges is the continuous and rapid growth of the web. The growth rate of the web is even more dramatic. According to [1], and [2], the size of the web has doubled in less than two years.

One of the challenges in the web is the interlinked nature of the web that sets it apart from many other collections. As shown in Figure 1.1, each page/document on the Web is represented as a node in a very large graph. The directed arcs connecting these nodes represent the hyperlinks between the documents.

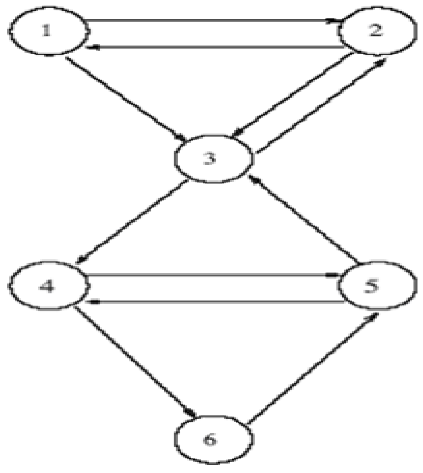

Figure 1.1 Hyperlink Structure of a 6-node sample Web

Several studies aim to understand how the web's linkage is structured and how that structure can be modeled [3], [4], [5], [6], [7]. One study, for example, suggests that the link structure of the web is somewhat like a "bow-tie" [3]. That is, about 28% of the pages constitute a strongly connected core (the center of the bow tie). About 22% form one of the tie's loops: these are pages that can be reached from the core but not vice versa. The other loop consists of 22% of the pages that can reach the core, but cannot be reached from it. The remaining nodes can neither reach the core nor can be reached from the core.

Search engines have main five components [8], as shown in Figure 1.2:

1- A crawling module for downloading web pages.
2- An indexing module for generating a lookup table for the downloaded pages.
3- A page repository for containing a local copy of the downloaded pages.
4- A query engine for fulfilling the user queries.
5- A page ranking module for sorting the search results.

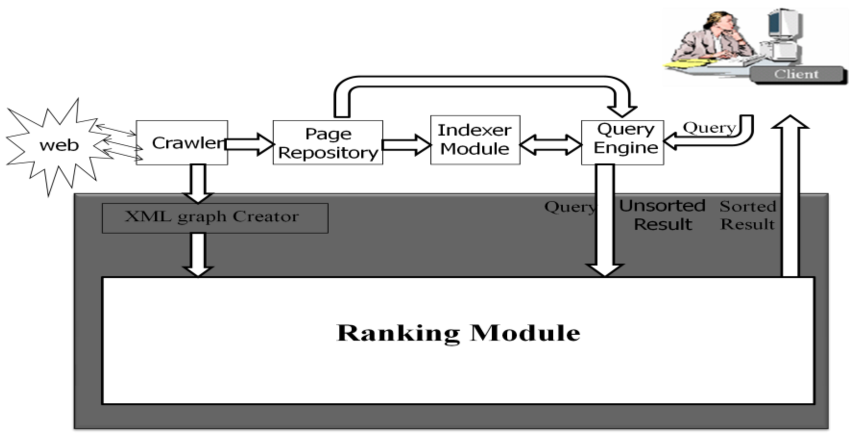

Figure 1.2: The general architecture of search engines.

Recent studies show that search engines play an increasingly important role in people's surfing of the web; when a user wants to look up information from the web, the user often goes to his favorite search engine, issues keyword queries, and clicks on the returned pages. Given the sheer quantity of information available on the web, the widespread use of search engines is not surprising. Because of the Web's size, and the fact that users typically only enter one or two keywords, result sets are usually very large [8]. The *ranking* module therefore has the task of sorting the results such that results near the top are the most likely ones to be what the user is looking for.

The following sections are organized as follows: section 1 provides a detailed description for each component in the search engine. Section 2 focuses on the ranking module. Section 3 highlights the motivation of the work presented in this thesis. A summarized explanation for the main objectives of the work presented in this thesis is given in section 4. And finally section 5 outlines the organization of the remaining parts of the thesis.

1.1 Preliminaries

The general architecture of the search engine is described in the following sub-sections.

1.1.1 The Crawling Module

Every engine relies on a crawler module to provide the grist for its operation (shown on the left in Figure 1.2). Crawlers are small programs that `browse' the web on the search engine's behalf, similarly to how a human user would follow links to reach different pages. The programs are given a starting set of URLs, whose pages they retrieve from the web. The crawlers extract URLs appearing in the retrieved pages, and give this information to the crawler control module. This module determines what links to visit next, and feeds the links to visit back to the crawlers. (Some of the functionality of the crawler control module may be implemented by the crawlers themselves.) The crawlers also pass the retrieved pages into a page repository. Crawlers continue visiting the web, until local resources, such as storage, are exhausted.

1.1.2 The Indexing Module

The indexing module is responsible for the process of generating a lookup table with all the URLs that point to pages containing a given word. This module is the most critical part of any search engine as it is considered its core. To index the web is quite a challenging task. Firstly, it requires huge resources and takes days to complete. Secondly, periodic crawling and rebuilding of the index is necessary because the contents of the web change rapidly and most incremental indexing-techniques perform poorly with huge changes. Finally, storage formats for the index must be carefully designed, for example, a compressed index may improve query performance, however, there is a tradeoff between this performance gain and the decompression overhead at query time.

The basic stages for indexing are as follows:

1- Extracting all the words from each document.
2- Removing stop words (e.g. an, and, by, for, of, the, etc...).
3- Normalizing words.
4- Eliminating very high and very low frequency terms.
5- Assigning a term weight using statistics.

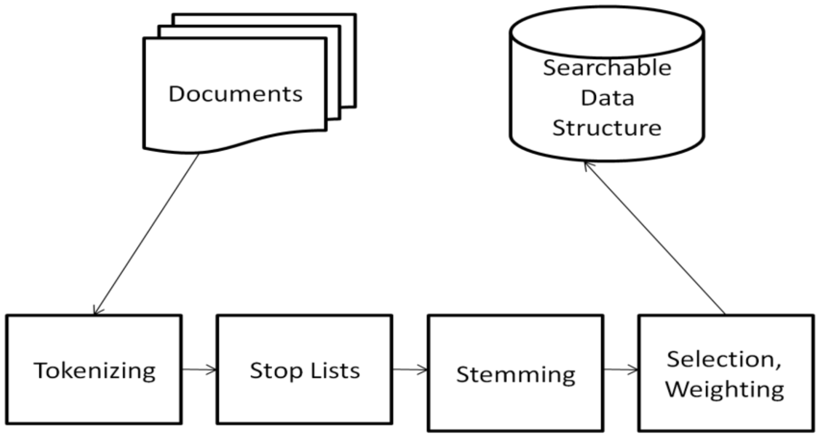

Figure 1.3: The basic stages of the indexer.

There are several types of indexes [8]. A link index represents a graph for the crawled portion of the web, with nodes (pages) and edges (hypertext links). Text index, however, is a lookup for identifying pages relevant to a query. The utility index provides access to pages of a given length, pages of a certain importance, or pages with some number of images in them.

Text index may be one of the following: inverted files, suffix arrays, and signature files. The inverted indexes are the most commonly used in indexing web pages. The inverted index structure is defined as follows:

- An inverted index over a collection of web pages consists of a set of inverted lists, one for each word (or index term).
- The inverted list for a term is a sorted list of locations where it appears in the collection.
- The posting is a pair of an index term *w*, and a corresponding location, *l*.
- Most text-indexes maintain a lexicon (a list of all the terms in the index with some term-level statistics).

BTrees, first presented in [9], are data structures that are commonly used in indexing large amount of data items. All leaves in a BTree are maintained on the same level. A very large number of items can be stored in a BTree of a small height. Furthermore, the maximum height of a BTree index determines the maximum number of accesses for a search, so the search time will be minimized if BTrees are used in indexing a very large amount of documents. It is argued that BTrees are ideal for indexing web pages [10], [11].

1.1.3 The Page Repository

The page repository is defined as follows: A cache of the visited pages that is maintained by a search engine beyond the time required to build the index. This cache allows serving out result pages quickly. It performs two basic functions: First, it is an interface for the crawler to store pages. Second, it provides an efficient access for the indexing module to retrieve the pages.

The challenges for page repository are:

1- Scalability: It must be distributed across a cluster of computers because of the huge size of the web.
2- Dual access modes: It must support two different access modes: Random access is used to quickly retrieve a specific page to the end user. Streaming access is used to receive the entire collection to the indexing module.
3- Large bulk updates: It must handle modifications. As new versions of pages are received, the space of old versions is reclaimed. Also conflicts between updating and accessing must be avoided.
4- Obsolete pages: It must have a mechanism for detecting and removing obsolete pages.

There are two policies for page distribution among repository nodes [8]: