

Efficient Techniques for Improving the Performance of Multimedia Search Engines

Saed Alqaraleh

Submitted to the
Institute of Graduate Studies and Research
in partial fulfilment of the requirements for the degree of

Doctor of Philosophy
in
Computer Engineering

Eastern Mediterranean University
November 2015
Gazimagusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Cem Tanova
Acting Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Doctor of Philosophy in Computer Engineering.

Prof. Dr. Işık Aybay
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Doctor of Philosophy in Computer Engineering.

Prof. Dr. Omar Ramadan
Supervisor

Examining Committee

-
1. Prof. Dr. Mehmet Ufuk Caglayan _____
 2. Prof. Dr. Oya Kalipsiz _____
 3. Prof. Dr. Omar Ramadan _____
 4. Assoc. Prof. Dr. Muhammed Salamah _____
 5. Asst. Prof. Dr. Gürcü Öz _____

ABSTRACT

The main objective of the work presented in this thesis is to improve the performance of multimedia search engines. The contributions of the work presented are: First, a watcher based crawler (WBC) that has the ability of crawling static and dynamic websites has been introduced. In this crawler, a watcher file, which can be uploaded to the websites servers, prepares a report that contains the addresses of the updated and the newly added webpages. The watcher file not only allows the crawlers to visit the updated and the newly webpages, but also solves the crawlers overlapping and communication problems. In addition, the proposed WBC is split into five units, where each unit is responsible for performing a specific crawling process, and this will increase both the crawling performance and the number of visited websites. The second contribution of this thesis is presenting a new re-ranking approach based on the multimedia files contents and some user specific actions. The proposed re-ranking scheme has the ability of working with all multimedia types: video, image, and audio. In addition, a group of multimedia descriptors that can be extracted from the file concurrently using multiple threads, will be used to describe accurately the multimedia file. Furthermore, the proposed re-ranking approach can show the most relevant files to the top of the query results, and can increase the percentage of the retrieved relevant files. Third, we have proposed an efficient scheme for eliminating duplicated files in multimedia query results, and finally, the performance of the query by example (QBE) has been enhanced to efficiently support all multimedia types.

Several experiments have been conducted to show the validity of the proposed approaches.

Keywords: Multimedia search engines, Information retrieval, Crawling algorithm, Re-ranking algorithm, Elimination of duplicated files, Query by Example.

ÖZ

Bu tezde sunulan yöntemin esas amacı çoklu ortam arama motorlarının performansını arttırmaktır. Burada sunulan işin getirilerinden ilki, izleyici tabanlı örün robotudur (Watcher Based Crawler, WBC). Bu robot, statik ve dinamik siteleri tarama özelliğine sahiptir. Önerilen sistemde, örün sunucularına atılabilen izleyici dosyası aracılığıyla güncellenen ve yeni eklenen web siteleri raporlanır. Bu izleyici dosyası örüntü robotunun yeni ve güncellenen sayfaları taramasını sağladığı gibi, robotların çakışma ve iletişim sorunlarını da çözmektedir. Buna ek olarak, WBC beş farklı birime ayrılmıştır. Bu birimler kendilerine özgü tarama işlemleri yaparak hem tarama performansını hem de ziyaret edilen sitelerin sayısını arttırmaktadır. İkinci olarak bu tezde, çoklu ortam dosya içeriklerine ve kullanıcı işlemlerine dayanan yeni bir sıralama yöntemi önerilmiştir. Önerilen sıralama yöntemi tüm çoklu ortam dosyalarıyla çalışabilmektedir (video, resim ve ses). Ek olarak, bir grup çoklu ortam tanımlayıcısı, çok sayıda iş parçası kullanılarak ayıklanıp çoklu ortam dosyalarını tanımlamak için kullanılmaktadır. Önerilen sıralama sistemi, aramayla ilgili kayıtların yukarıda çıkmasını arttırdığı gibi, bulunan dosyaların konuyla ilgili olma oranını da arttırmaktadır. Bu Çalışmadak üçüncü katkı ise, arama sonuçları listesinde bulunan aynı sonuçları ayıklayan verimli bir sistem olmasıdır. Son olarak, tüm çoklu ortam dosyalarını verimli bir şekilde desteklemek için, örnekle sorgulama (Query by Example, QBE) yöntemi kullanılmıştır. Oluşturulan bu sistemin doğrulanması için, çeşitli deneyler yapılmıştır.

Anahtar kelimeler: Çoklu ortam arama motorları, Bilgi çağırma, Bilgi erişim sistemi, tarama algoritması, Sıralama algoritması, Eş dosyaların elenmesi, Örnekle sorgulama, Çoklu ortam arama motorları.

DEDICATION

To My Family
(Especially to my Father and my Mother)

أهداء الى كافة اعضاء عائلتي
وبالخصوص الى أبي وأمي

ACKNOWLEDGMENT

In the name of Allah the most Merciful and Beneficent

First and foremost all the praises and thanks to Allah, the almighty Merciful, the greatest of all, who ultimately I depend on for the guidance in my whole life.

I would like to thank Prof. Dr. Omar Ramadan, for the guidance and support. I would also like to thank him for his continuous encouragement, patience and for sharing his knowledge, as well as his effort in proofreading the drafts, are greatly appreciated. Without any doubt, his appreciated supervision, lead me to be in this position.

Besides my advisor, I would like to thank and express my honour for Prof. Dr. Mehmet Ufuk Caglayan and Prof. Dr. Oya Kalipsiz for being a member in my thesis committee. In addition, a big thanks for Assoc. Prof. Dr. Muhammed Salamah and Asst. Prof. Dr. Gürcü Öz for being a part of this whole journey and for their encouragement and insightful comments. I must also acknowledge Asst. Prof. Dr. Yıltan Bitirim for his valuable suggestions and comments that greatly improved this work.

My family, I can't find words that express my gratitude, without you this work wouldn't be possible. I owe a huge debt of gratitude to my parents for their unconditional support and all they have done for me. My dad, mum, grandfathers, grandmothers, sisters, brothers, aunts, uncles, and all my family members, without

your prayers I wouldn't be here. I hope that I have achieved the dream that you had for me.

I am really indebted to wife and my son (Alwaleed) for their understanding, patience, and unconditional love. I would like also to thank my wife for the sleepless nights and for the support in the moments that I had difficulties.

Last but not least, I would like to express my sincere appreciation to my second family, all members of Computer Engineering Department and all my friends for their helpful attitude and constant support.

TABLE OF CONTENTS

ABSTRACT	iii
ÖZ	v
DEDICATION	vii
ACKNOWLEDGMENT	viii
1 INTRODUCTION	1
2 OVERVIEW OF INTERNET SEARCH ENGINES	5
2.1 Introduction.....	5
2.2 Mechanism of the Conventional Search Engines	6
2.3 Recent Developments in Search Engines	11
3 THE PROPOSED WATCHER BASED CRAWLER.....	12
3.1 Introduction.....	12
3.2 Related Works	12
3.3 Main Problems in the Existing Crawling Techniques	15
3.4 The Proposed Watcher Based Crawler Structure	18
3.4.1 The Watcher File	18
3.4.1.1 Mechanism of Building the Watcher’s Report.....	19
3.4.1.2 Watcher File Setup	25
3.4.2 WBC-Server Design.....	25
3.4.3 WBC Complexity Analysis	32
3.4.4 WBC Properties.....	32
4 THE PROPOSED RE-RANKING APPROACH	35
4.1 Introduction.....	35
4.2 Related Works	35
4.3 The Proposed Re-ranking Structure	37

4.3.1 Offline Operations.....	38
4.3.1.1 Pre-processing Operations.....	39
4.3.1.2 Extract File Features.....	41
4.3.2 Online Operations	46
5 THE PROPOSED ELIMINATION APPROACH FOR DUPLICATED MULTIMEDIA FILES	49
5.1 Introduction.....	49
5.2 The Proposed Elimination Structure.....	50
5.2.1 Hash Algorithms and Feature Extraction Techniques.....	53
5.2.1.1 Hash Algorithms.....	53
5.2.1.2 Feature Extraction	54
5.2.2 Mechanisms of Multimedia Files Comparison Process	55
5.2.3 Parallel Implementation of the Proposed Elimination Scheme.....	56
6 THE ENHANCED QUERY BY EXAMPLE	57
6.1 Introduction.....	57
6.2 The Enhanced QBE Structure.....	59
6.2.1 Clustering Techniques.....	60
6.2.2 Parallel Implementation of the Proposed QBE	64
6.2.3 Elimination of Duplicated Multimedia Files.....	66
6.3 Query Processing Mechanism	67
6.3.1 User Options for the Queries Using the Proposed QBE	69
7 EXPERIMENTAL STUDY	70
7.1 Crawler Technique Performance	70
7.2 Re-ranking Technique Performance	81
7.3 Elimination Technique Performance	94
7.3.1 Image Database Processing.....	94

7.3.2 Video and Audio Databases Processing.....	96
7.4 QBE Technique Performance	104
8 CONCLUSIONS AND FUTURE WORKS.....	112
REFERENCES.....	115

LIST OF TABLES

Table 6.1: Total number of comparison of the enhanced QBE as compared with A, B scenarios.....	66
Table 7.1: The number of updated pages for different websites recorded in seven days.	71
Table 7.2: The frequency of downloading specific webpages in three days by running fifty parallel crawlers.	73
Table 7.3: The watcher file requirements.	74
Table 7.4: Number of downloaded distinct webpages using the proposed WBC and WEB-SAILOR [28].	75
Table 7.5: Number of downloaded webpages using the proposed WBC, Apache Nutch [119], and Scrapy [120].....	76
Table 7.6: Number of comments for specific YouTube videos reported in [121] and downloaded using the proposed WBC.....	80
Table 7.7: The 10-Precision of the SCD, EHD, JCD, and the proposed approach for re-ranking image and video queries.	82
Table 7.8: The position of first five relevant files for specific queries obtained using Google, and Yahoo search engines.	84
Table 7.9: The position of first ten relevant files for specific queries using JCD, and the proposed re-ranking approach (PRA).	85
Table 7.10: The 10-Precision of the proposed approach with and without the pre-processing operation for re-ranking image, video and audio queries.	87
Table 7.11: 10-Precision of “Amazon MP3”, and the proposed approach using a single, and three dynamic signatures.	88
Table 7.12: R-Precision of the developed re-ranking approach for a sample of the image queries.	90

Table 7.13: R-Precision of the developed re-ranking approach for a sample of re-ranking video queries.	91
Table 7.14: R-Precision of the developed re-ranking approach for a sample of re-ranking audio queries.	91
Table 7.15: Execution time for updating a database using the proposed elimination scheme.	100
Table 7.16: The percentage of relevant and duplicated files for Google search engine with and without using the proposed elimination scheme.	101
Table 7.17: The percentage of relevant and duplicated files for Yahoo search engine with and without using the proposed elimination scheme.	102
Table 7.18: The percentage of relevant and duplicated files for Bing search engine with and without using the proposed elimination scheme.	103
Table 7.19: Percentage of relevant files for some video queries using K-means, subtractive, spectral, hierarchical, and neural network algorithms.	105
Table 7.20: Effect of clusters number on the percentage of relevant files for some video queries using the proposed ensemble system.	106
Table 7.21: Execution time using the sequential QBE scheme, sequential QBE with clustering scheme and parallel QBE with clustering scheme for different number of multimedia files.	107
Table 7.22: Comparison between Google QBE and the enhanced QBE for image queries.	108
Table 7.23: The efficiency of the enhanced QBE approach for videos/audios.	110
Table 7.24: Comparison between the Google, Yahoo and Bing text based search engines versus the enhanced QBE.	111

LIST OF FIGURES

Figure 2.1: The conventional search engines mechanism.....	5
Figure 3.1: The WBC structure.....	18
Figure 3.2: Flowchart of the developed watcher file for adding the triggering paths of onload, onclick, ondblclick and onmouseover events to the watcher report.	22
Figure 3.3: Flowchart of the developed watcher file for adding the updated static pages to the report.....	24
Figure 3.4: Flowchart of the developed crawler unit for processing static webpages.	28
Figure 3.5: Flowchart for processing the dynamic pages by the AJAX unit. *This process is done according to Algorithm 3.1.....	30
Figure 4.1: The structure of the proposed re-ranking approach.....	38
Figure 5.1: The flowchart of eliminating the duplication of the multimedia files during creating and/or adding new files to multimedia database(s).	52
Figure 6.1: The structure of the enhanced QBE.....	60
Figure 6.2: The flowchart to define the number of clusters, where z is number of clusters.....	63
Figure 6.3: The flowchart of parallel implementation of part II of the proposed QBE, using M threads, for Z different clusters.....	65
Figure 6.4: The flowchart of the query process unit in the proposed QBE.	68
Figure 7.1: The total number of downloaded webpages using dependent, seed-server, independent strategies, and the developed WBC in three days.	74
Figure 7.2: The required time for re-visiting “www.hkjtoday.com” website using the crawlers of WBC, uniform, and proportional by rank and by top N level.....	77

Figure 7.3: The percentage of downloading updated pages in “www.hkjtoday.com” website using the crawlers of WBC, uniform, and proportional by rank and by top N levels.	77
Figure 7.4: Number of dynamic pages processed in 10 minutes for (a) one crawler, (b) two crawlers, (c) three crawlers, and (d) four crawlers.	79
Figure 7.5: The R-Precision for the image queries.	89
Figure 7.6: The R-Precision for the video queries.	89
Figure 7.7: The R-Precision for the audio queries.	90
Figure 7.8: R-Precision of the developed re-ranking approach and the approach presented in [34] for specific image queries.	92
Figure 7.9: R-Precision of the developed re-ranking approach and the approach presented in [41] for specific video queries.	93
Figure 7.10: Execution time versus number of images for the MD5 and Bit-wise algorithms.	95
Figure 7.11: Execution time versus number of images for 4, 8, 12, and 16 processes as obtained by using the parallel implementation of the MD5 algorithm.	95
Figure 7.12: Execution time versus number of video and audio files for 4, 8, 12, and 16 processes as obtained by using the parallel implementation of the MD5.	97
Figure 7.13: Execution time required for creating video/audio databases using the low level extraction and the MD5 hashing algorithm.	98
Figure 7.14: Execution time required for the comparison process by using the low level extraction and the MD5 hashing algorithm.	98
Figure 7.15: Total execution time versus number of file using the low level extraction and the MD5 hashing algorithm.	99

Chapter 1

INTRODUCTION

In the early days of the Web, the limited amount of information that was available leads the user to find websites and relevant information manually [1]. In the 1990 century, the number of websites, documents and resources had increased astronomically. This makes the process of finding certain information manually difficult and sometimes impossible. As a result, web search engines were introduced [2]. Web search engines are websites that designed to help users in getting specific information on the World Wide Web (WWW). Nowadays, most search engines use the crawler techniques [3-5] to collect website's information such as, site's meta tags, keywords, multimedia files, etc. Crawler, which is also known as a spider or robot, is a software that visits the websites in a routinely manner. The main aims of the crawler are to find new web objects, such as new webpages, multimedia files, articles, etc., and to observe changes in previously indexed web objects. The crawler will return all extracted information back to the search engine central server to be indexed and saved in the databases. Databases mainly contain keywords, URL addresses, copy of the webpages, multimedia files and other related information. When the user executes a query, the search engine finds the most relevant information, and by a specific ranking algorithm, the results will be ordered and shown to the user.

Nowadays, multimedia files are among the most important materials on the Internet. In the last few years, the number of published multimedia files has grown considerably, and several developments in the field of accessing the multimedia have been introduced. However, even for the recently state-of-the-art methods and applications based on accessing multimedia files on the Internet, we still have challenging problems. In the following, the main problems related to multimedia searching are described.

- 1) Most of current crawlers use the conventional crawling techniques that can't detect the updated pages online [3-5] and this will necessitate the crawlers to download all the webpages. This means that the conventional crawlers spend most of their working time in visiting un-updated pages. It is worth mentioning that webpages can be categorized into static which are the webpages that allocated in the website's server and delivered to the user exactly as being stored on the server, and dynamic webpages, which use the AJAX techniques [6], and in most cases, are not allocated on the server, i.e., generated dynamically. Therefore, the conventional crawling techniques can download only the pages that are allocated on the website server, and they are inefficient when dealing with AJAX pages, as they can't index the website's dynamic information [7-9].
- 2) Most search engine's multimedia databases are still created based on the multimedia metadata and the surrounding text. This technique does not pay attention to the contents of the file itself, and sometimes there may be no relation between the contents of the multimedia files and its metadata and/or the surrounding text. This may lead the outcomes of multimedia queries to contain a large number of irrelevant files.

- 3) In the 1970's, a new searching techniques known as query by example (QBE) that can be used to find files that are similar to what the user already has, was introduced [10]. Although, its performance is good for image files, it still suffers problems for the other multimedia files such as videos and audios.
- 4) Last but not least, it has been found in [11-14] that around 40% of the pages and multimedia files on the web are duplicated.

The main objective of the work presented in this thesis is to solve the above mentioned problems and improve the overall performance of existing multimedia search engines as listed below.

- I. A watcher based crawler (WBC) [15] that has the ability of crawling static and dynamic websites has been presented. In the proposed crawler, a watcher file, which can be uploaded to the websites servers, prepares a report that contains the addresses of the updated and the newly added webpages. The watcher file not only allows the crawlers to visit the updated and newly webpages, but also solves the crawlers overlapping and communication problems. In addition, the proposed WBC is split into five units, where each unit is responsible for performing a specific crawling process, and this will increase both the crawling performance and the number of visited websites.
- II. A new re-ranking approach based on the multimedia contents and some user specific actions is introduced. This approach has the ability to work with all multimedia types: video, image, and audio. In addition, a group of multimedia descriptors will be extracted from the file concurrently using multiple threads to describe accurately the file itself.

- III. Elimination of duplicated files in multimedia query results has been introduced [14] by using some techniques such as hashing algorithms [16] and feature extraction [17].
- IV. Finally, QBE approach has been enhanced [18, 19] by adapting some techniques like dynamic descriptors and clustering.

Several experiments have been conducted to study the performance of the above introduced techniques. It has been observed that the proposed WBC increases the number of uniquely visited static and dynamic websites as compared with the existing crawling techniques. In addition, the proposed re-ranking approach shows the most relevant files to the top of the query results, and increases the percentage of the retrieved relevant files. Furthermore, we have successfully managed to eliminate duplicated files completely, and finally the QBE was enhanced to support all multimedia types with good accuracy levels.

The remaining of the thesis is organized as follows: Chapter 2 provides the mechanism of search engines. Chapter 3 presents the developed WBC. The proposed re-ranking approach is described in Chapter 4. Chapter 5 explains the proposed methodologies for eliminating the duplicated files in multimedia search engines. Chapter 6 presents the enhanced QBE approach. Experimental studies are presented in Chapter 7, and finally, conclusions are given in Chapter 8.

Chapter 2

OVERVIEW OF INTERNET SEARCH ENGINES

2.1 Introduction

Web search engines are websites designed to help users in getting specific information on the WWW. In general, search engines use crawler techniques to collect website's information such as site's meta tags, keywords, multimedia files, etc. Then, the collected information will be analyzed to build the databases. When the user executes a query, the search engine finds the most relevant webpages, and by specific ranking algorithms [20], the results will be ordered and shown to the user. The mechanism of the convention search engines is shown in Figure 2.1.

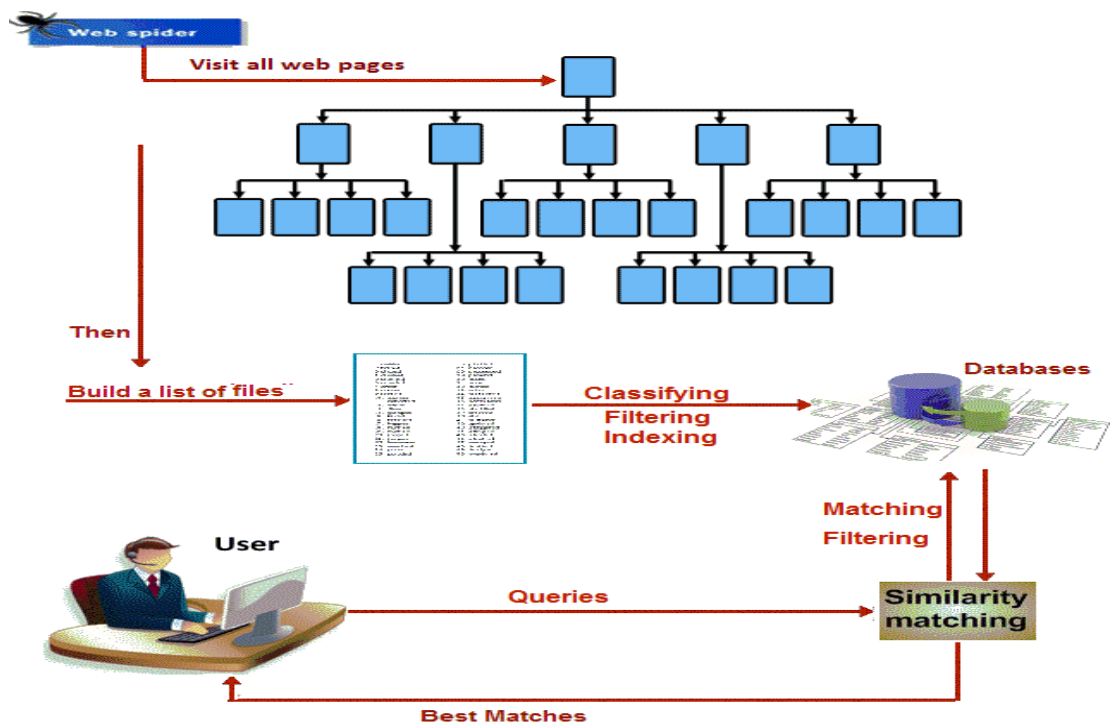


Figure 2.1: The conventional search engines mechanism.

2.2 Mechanism of the Conventional Search Engines

Based on Figure 2.1 the mechanism of the conventional search engines includes five steps. The details of these steps are explained below.

2.2.1 Crawling Process

A crawler, also known as spider or robot, is a software that visits all websites over the Internet, downloads the web documents and stores the collected documents on the search engine servers [3-5],[7-9]. In general, the mechanism of the crawler can be summarized as follows:

- 1) The crawler starts crawling based on a set of URLs, i.e., URLs frontier.
- 2) The crawler downloads a page, extracts its URLs and inserts these URLs into a queue. It is worth mentioning that the contents of the queue will be added in a sorted manner to the crawler's URLs frontier.
- 3) Save the downloaded page in the search engine database(s).
- 4) Steps 1-3 will be repeated for the next URL, and the crawling process can be stopped based on specific criteria, such as the frontier is empty and/or a predefined stopping time specified by the crawler administrator, etc.

The primary goals of the crawler are a) finding new web objects, and b) observing changes in previously indexed web objects. To achieve the first goal, the crawler has to visit as many websites as possible, and to achieve the second one, the crawler has to maintain the freshness of the previously visited websites, which can be achieved by re-visiting such websites in a routinely manner. In the following, the most frequently used re-visiting policies are summarized:

- 1) Uniform policy: In this policy, the entire websites pages will be downloaded at each visit [21-25]. Although, this approach enriches the databases, it requires a large processing time [21-25].
- 2) Proportional policy: This policy is performed in different ways. In the following the most frequent used ones are listed:
 - a) Download only the pages that have a rank more than a threshold value specified by the crawler administrator [21-23]. The rank of a page is based on many factors such as the importance and the frequency of updating the page.
 - b) Download the webpages allocated in the top N levels of each website. In general, this type of proportional policy, which is based on breadth first algorithm [21, 26], involves visiting the main page (root) of the website, and downloading only the pages that have URL links or allocated in the top N levels [21, 26]. This helps the crawler to avoid exploring too deeply into any visited website [27].

It is worth mentioning that the time required for re-visiting a website for the proportional policy is significantly less than the uniform policy. On the other hand, the proportional policy may ignore visiting the new webpages, as their rank is initially low, and it may also ignore the updated webpages which are not allocated on top N levels.

Based on the fact that there are a huge number of websites, parallel techniques were used to speed up the crawling operations. Parallel crawlers can be static or dynamic [30], which are described as below.

1) Static crawlers

In this case, the websites are divided and assigned statically to each crawler, and there is no controller to co-ordinate the activities of individual crawlers. This type of crawlers can be categorized into two groups:

- a) Dependent crawlers: In this case, all running crawlers have to communicate with each other, and this increases the time of the crawling process, which degrades the overall performance of the crawling system. For example, if we have 3 hundred crawlers working at the same time, and when any crawler visits a webpage, this crawler has to communicate with the other 299 crawlers to make sure that they didn't visit this webpage.
- b) Independent crawlers: In this case, each of the working crawlers has its own decision without communicating other. The main problem with this type of crawlers is the overlapping issue, which means more than one crawler may visit and download the same webpage.

2) Dynamic crawlers

In the dynamic crawlers, a controller is used to partition the websites into groups and assigns each group to a specific crawler. As in the static crawlers case, overlapping problem may occur with this type of crawlers.

In [3-5], [28], the main introduced techniques that can be used for solving overlapping problem are summarized as below.

- 1) The crawler will discard the URLs that are not in its URLs frontier and continue to crawl on its own partition erasing any possibility of overlap [3-

- 5]. But, in this case, many important URLs will be lost, and this will reduce the quality of the query results [3-5].
- 2) The running crawlers have to communicate with the server to decide the crawling decisions [28]. Although this approach eliminates the communication between the running crawlers, an extra time delay is required for communicating with the server to decide on visiting the websites.

2.2.2 Indexing Process

The second major step of the conventional search engines mechanism is the indexing process. This step starts by analysing the information collected by the crawlers. Then, the crawled pages and its information such as, the keywords, the website's address and multimedia files, will be saved in the databases. It is worth mentioning that, as the contents of the websites change frequently, the search engines must keep maintaining and updating all database(s) contents.

2.2.3 Searching Process

When a user requests a certain information or webpage by writing keyword(s), i.e., query, the search engines, in most cases, use Boolean operations, such as “and”, “or”, etc. to control the relation between the words in the queries. In addition, a spell checking process that allocate misspelled words and notify the user of the misspellings has been introduced, to increase the chance of detecting the required information and to improve the search results.

2.2.4 Database Searching

When a query is submitted, the search engine will post a request to the databases to find the related websites. In general, search engines perform statistical analysis on

the indexed pages to find the most relevant ones. In last few years, search engines start to use “caching” [29] to reduce the processing time while searching for common queries. In this case, the search engine returns the result from the cache without checking the databases [29].

2.2.5 Page Ranking

In this step, the search engine will rank the websites in a list that will be displayed as the query result. Basically, the mechanism of ranking the websites is categorized into the following two categories:

- 1) Query independent ranking scheme [30-32]: This scheme ranks the importance of websites based on some factors like the number of hits, the keywords, the website meta-tags, its contents, etc.
- 2) Query dependent ranking scheme [32-34]: In most cases, it is a distance based scheme that ranks the files by calculating their distance to the query.

Then, the website with a high rank will be shown to the top of the listed results. On the other hand, the above ranking mechanisms do not pay attention to the contents of the requested file itself. This is unfair with multimedia files which are an important part of the web contents, as both text and multimedia files contents can contain useful information that should be used in ranking the websites. This is because in some cases, there may be no relation between the contents of the multimedia files and the surrounding text, and this will lead the outcomes of query to contain a large number of irrelevant files. Hence, re-ranking techniques have been introduced to improve the quality of the retrieved information [35-41]. In general, the mechanism of the re-ranking is based on re-order the query’s results based on their relevance.

2.3 Recent Developments in Search Engines

In the last few years, the number of published multimedia files has grown considerably, and several developments in the field of accessing the multimedia have been introduced [42- 48]. In [42], an image retrieval system working on extracting information from files (content based retrieval) has been developed. In [43], a new mechanism, which uses a hybrid method that combines ontology and content based methods, was presented for effective searching through multimedia contents. A new search engine for the scientific researches and learning purposes has been presented in [44]. In this search engine, several score functions have been introduced to improve the order of the query's relevant pages. In addition, an anchor text analyser that analyses pages that may or may not contain the query terms to decide if the pages are relevant. Furthermore, the crawler of this search engine has the ability of finding the priority for URLs queue, and balancing the load of the crawling process. In [45], a semantic approach that presents the multimedia documents based on conceptual neighbourhood graphs has been proposed. In [46], a 3D model retrieval technique based on 3D fractional Fourier transform has been introduced to improve the searching outcomes. In [47], a hybrid search engine framework based on "historical and present sampling values" was presented. This search engine supports three kinds of search conditions: keyword-based, spatial-temporal and value-based. In [48], a group of researchers have designed collection of tools named as SpidersRUs, which can be used in building crawling, indexing and searching functions. Finally, in addition to the above techniques, it is important to note that most of the information about the commercial search engines such as Google and Bing are kept hidden as business secrets, and there are very few documents about the mechanisms of these engines.

Chapter 3

THE PROPOSED WATCHER BASED CRAWLER

3.1 Introduction

In the last decade, the number of websites, documents and resources had increased astronomically. As of February 2015, the total number of the published websites on the Internet was over 1.25 billion [49]. Up to date, the number of websites that can be processed by the conventional crawlers compared with the current huge number of published websites is still limited. To speed up the conventional crawlers, a very large number of terminals must be used [3-5]. This increases both the cost and the complexity of the crawling system. Therefore, improving the crawling techniques have been and continue to be an important issue.

3.2 Related Works

In the last few years, several developments in the web crawler field have been introduced. In [28], a dynamic parallel web crawler based on client-server model, named as WEB-SAILOR, has been introduced. This approach eliminates the communication between the running crawlers by introducing a seed-server which is responsible for the crawling decisions. In [50], a new crawler, named as DCrawler, has been implemented. In this crawler, a new assignment function is used for partitioning the domain between the crawlers. In [51], a parallel web crawler based on the cluster environment was presented. In this approach, a new distributed controller pattern and dynamic assignment structure were used. In [52], a scalable

web crawling system, named as WEBTracker, has been proposed to increase the number of visited pages by making use of distributed environment.

In addition, topic specific crawlers were introduced for specific topic search engines, where the databases contain websites that are related to a specific topic(s) only [53, 54]. These crawlers, which are also known as focused crawlers, download only webpages that are relevant to a pre-defined topic(s). In [53], a focused web crawler which calculates the prediction score for the unvisited URL's based on the webpage hierarchy and the text semantic similarity was introduced. In [54], a multiple specific domains search engine has been developed. The main problem in the developed crawlers of [53] and [54] is that they need to visit all published worldwide websites and this will increase the required time for updating the databases and showing the new information on query results.

In last ten years, rich Internet applications (RIAs) [6], that enhance and support the accessibility of scripted and dynamic contents, become more and more popular. AJAX, which is a group of interrelated techniques that can be used on the client side to create asynchronous web applications, can be considered as the most popular technique used in RIAs [6]. Hence, dynamic indexing crawlers were also introduced. In [7], an AJAX crawler that crawls dynamic webpages has been developed. In [8], an ontology based web crawler that can download the information in dynamic pages has been proposed. In [9], a new crawler, which extracts the information in dynamic pages by analysing java script language, was introduced. In [55], a new crawling methodology, named as model-based crawling, was introduced to design efficient crawling strategies for RIAs. Although current AJAX based crawlers can extract a

promising percentage of dynamic pages information, it is important to note that this process still requires a large processing time and this will slow down the process of extracting dynamic data [7-9].

From the above survey, it can be concluded that the webpages can be categorized into the following two categories: static and dynamic webpages. The details of these categories are described below.

- a) In the static case, webpages are allocated in the website's server and delivered to the user exactly as stored on the server [56].
- b) In the dynamic case, the webpages use AJAX techniques. In addition, triggering AJAX events may dynamically introduce new pages, known as states, and in most cases these pages are not allocated on the server [6-9].

It should be noted that the process of updating static pages can be done by editing and changing the contents of the file offline. On the other hand, the content of AJAX pages can be generated and updated dynamically (online) without reloading the whole page or changing the URL.

3.3 Main Problems in the Existing Crawling Techniques

In this section, the main crawler challenging problems are summarized:

- 1) Most of current crawler techniques can't detect the updated pages online [3-5], and this will necessitate the crawler to download all the webpages, and hence, increases the crawling processing time, the Internet traffic and the bandwidth consumption [57, 58].
- 2) Overlapping problem where more than one crawler process the same webpage.
- 3) Up to our knowledge, most of the crawling techniques require communications between the running crawlers, and this increases the crawling processing time and requires high quality networks [3-5, 28, 42, 50].
- 4) The conventional crawlers work is based on the URLs, and it download only the pages that are allocated on the website server. Therefore, conventional crawlers are inefficient when dealing with AJAX pages, as they can't index the website's dynamic information [7-9].

In addition to the above static crawling problems, AJAX crawling techniques are still suffering from several challenging problems such as:

- 1) Identifying webpage's states: In some cases, in order to identify the page states the AJAX events need to be triggered, and this may change the content of the corresponding page without changing the page URL, and such page will be recognized as one of the page's states.
- 2) As AJAX technique contains many events, triggering all page events will lead to a very large number of states. In some cases, more than one event may lead to the same state, and this will be considered as duplicated state. For

example, by clicking the “next” tab in page ($i-1$) and clicking the “previous” tab in page ($i+1$), will lead to the same page (i), i.e., the same state [62].

As it has been mentioned before, the conventional crawler techniques necessitate downloading all website pages to find the updated ones, and this will increase the Internet traffic and the bandwidth consumption. It has been found that approximately 40% of the current Internet traffic, bandwidth consumption and web requests are due to search engine’s crawlers [57, 58]. To solve this issue, mobile crawlers [58, 59] and sitemaps based crawlers [59-61] were introduced. The details of these two techniques are explained below:

A. Mobile crawlers:

Unlike the conventional crawlers, the mobile crawlers go through the servers of the URLs in its frontier to detect and store in its memory the required data such as the updated pages. This mechanism reduces the amount of data transferred over the network and therefore, decrease the network load caused by the crawlers. However, mobile crawling techniques are not wildly used due to the following problems:

- 1) Mobile crawlers occupy large portion of the visited website resources such as memory, the network bandwidth, CPU cycles, etc. [57].
- 2) Due to security reasons, the remote system in most cases will not allow the mobile crawlers to reside in its memory, and may recognize it as viruses.

B. Sitemaps based crawlers:

Sitemap file is an XML file that contains a list of the URLs of website pages with additional metadata such as: *loc*, which is a required field representing the URL of the webpage, and *lastmod*, which is an optional field representing the last time the webpage of the URL was updated. Recently, the websites start providing sitemap(s)

to the users for easier navigation. Sitemaps based crawlers use the sitemap(s) to find the updated and the new webpages. However, this mechanism suffers from the following problems:

- 1) Nowadays, many websites do not have sitemap, and therefore, the web crawler has to follow the traditional way of crawling.
- 2) Recently, a group of websites and softwares have been developed to create the website's sitemaps automatically. However, when any of the site pages is updated, the system administrator has to update the pages metadata such as *lastmod* manually [58, 59]. This makes the process of updating the sitemap, especially for larger websites, difficult and time consuming.

The main objective of the work presented in this chapter is to solve the above mentioned problems. A watcher based crawler (WBC) that has the ability of crawling static and dynamic websites has been presented. In the proposed crawler, a watcher file, which can be uploaded to the websites servers, prepares a report that contains the addresses of the updated and the newly added webpages. The watcher file not only allows the crawlers to visit the updated and newly webpages, but also solves the crawlers overlapping and communication problems. In addition, the proposed WBC is split into five units, where each unit is responsible for performing a specific crawling process, and this will increase both the crawling performance and the number of visited websites. Several experiments have been conducted and it has been observed that the proposed WBC increases the number of uniquely visited static and dynamic websites as compared with the existing crawling techniques.

3.4 The Proposed Watcher Based Crawler Structure

The developed WBC consists of two main parts: the watcher file, and the WBC server. The structure of these parts is shown in Figure 3.1, and the details of these parts are summarized below:

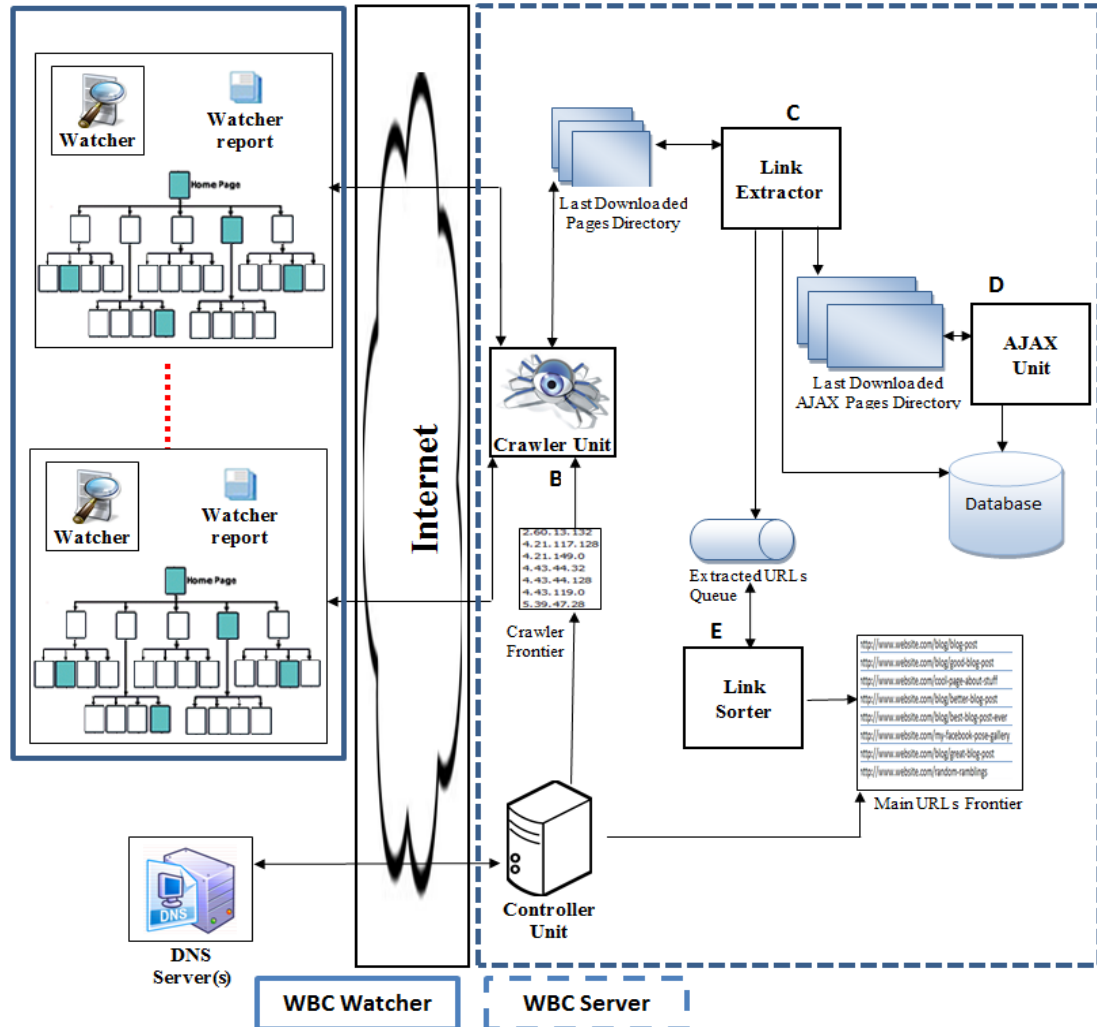


Figure 3.1: The WBC structure.

3.4.1 The Watcher File

In the proposed WBC, the watcher file, which will be uploaded to the websites server, prepares a report that contains only the updated and the newly added webpages. The watcher file is small in size which does not require any specific

requirement on the web server and it will not affect its performance. The main advantage of the watcher file is that it allows the crawlers to visit only the updated and the newly added pages. In addition, it solves crawlers overlapping and communication problems by introducing a flag in the watcher report, which will be set to 1 by the watcher file when a crawler processes that website. In this case, other copies of WBC will not visit any website whose flag is set. Hence, there will be no need for communication between the running crawlers.

3.4.1.1 Mechanism of Building the Watcher's Report

The mechanism of building the watcher report is performed using the following monitoring and ranking functions:

- 1) Monitoring function: This function keep track of the website directories and detects the updated and the newly added pages. In the case of updating static pages, the ranking function will be called to rank and add the pages URLs in the appropriate position in the report. In the case of dynamic pages, the monitoring function detects and adds the AJAX events including its triggering path into the watcher report as shown in Figure 3.2. The format of adding the dynamic page information is:

Dynamic page URL, Event(s), Source element(s), Target element, Value.
Triggering path

where *event(s)* is the java script event that will be triggered, *Source element(s)* represents the corresponding HTML object, *Target element* is the object whose information will be updated, and *Value* is the new value that will be assigned to the target or the name of the function that will be called. For instance, a dynamic page event can be represented in the report as follows:

www.test.com/main.asp, onclick, div id="next", text. innerHTML, Updateinfo()
Dynamic page URL Event Source Target Value

As it has been mentioned before, one of AJAX crawling problems is that same state may be retrieved multiple times. The following two scenarios illustrate such problem: First, in some cases, more than one HTML object in an AJAX page may contain exactly the same event and its triggering path. For instance, by triggering the following two reported events, the same state will be produced:

www.test.com/main.asp, onclick, div id="next", text. innerHTML, Updateinfo()
Dynamic page URL Event Source Target Value
www.test.com/main.asp, onclick, div id="previous", text. innerHTML, Updateinfo()
Dynamic page URL Event Source Target Value

The second scenario is that multiple events may have the same triggering path that produces the same state. For instance, the following events produce the same state:

www.test.com/main.asp, onclick, div id="next", text. innerHTML, Updateinfo()
Dynamic page URL Event Source Target Value
www.test.com/main.asp, onload, body id="next", text. innerHTML, Updateinfo()
Dynamic page URL Event Source Target Value

To overcome this problem, the monitoring function has the ability of detecting and combining the information of all similar events in one field. This allows the WBC to trigger only one of the similar events. For instance, a group of combined similar events is represented in the report as follows:

```
www.test.com/main.asp, {onclick, onload}, {button id="move",div id="next"},  
text.innerHTML, Updateinfo()
```

It is worth mentioning the followings: First, the watcher file will consider only the most important JavaScript events: onload, onclick, ondblclick and onmouseover [62]. Second, the WBC ignores the pages that require some database queries, which necessitates the user intervention to fill some forms, such as login pages. We believe that this type of information is private and shouldn't be indexed by crawlers.

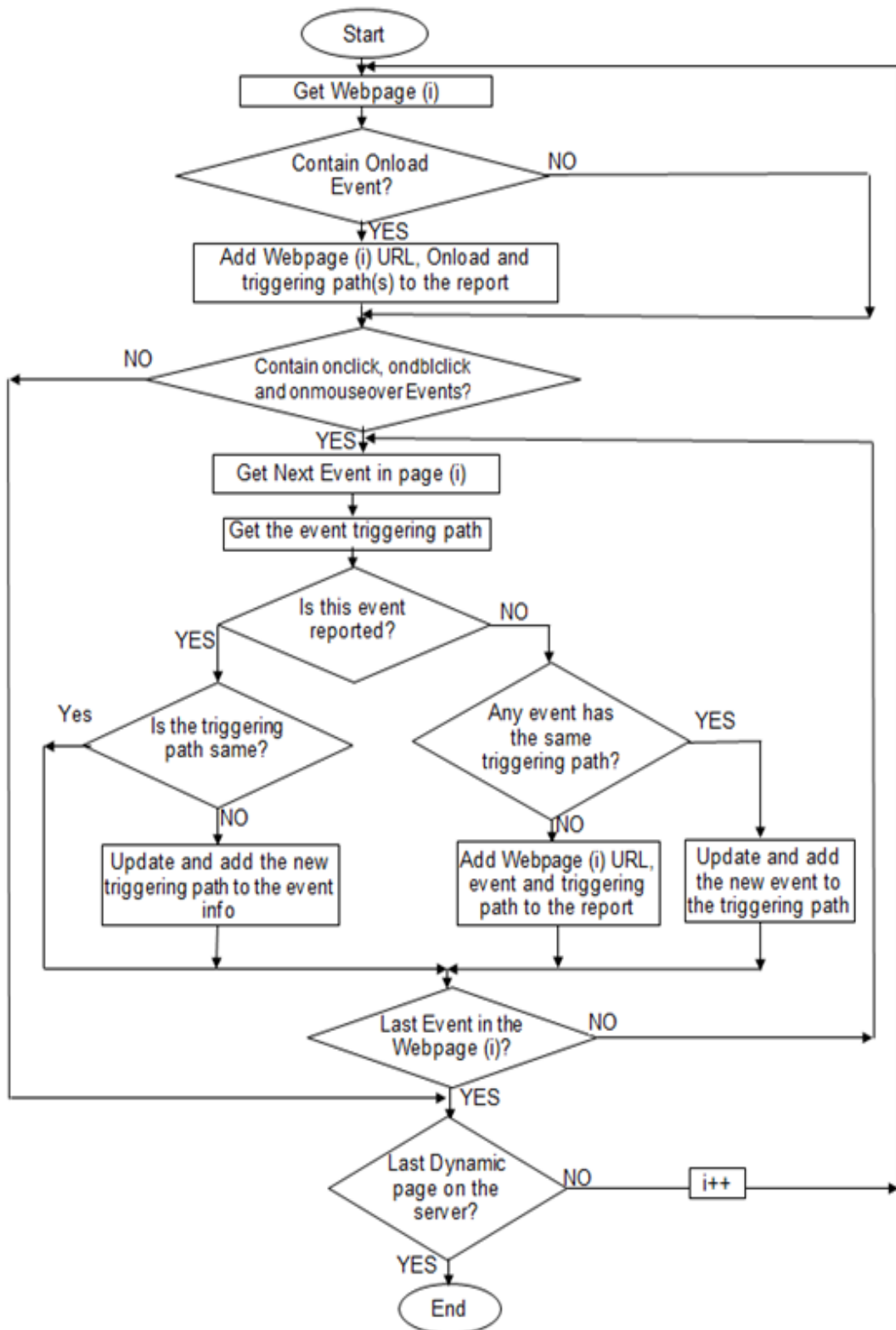


Figure 3.2: Flowchart of the developed watcher file for adding the triggering paths of onload, onclick, ondblclick and onmouseover events to the watcher report.

2) Ranking function: Based on Figure 3.3, this function is responsible for ranking and adding the URLs of the updated and the new static pages into the appropriate position of the watcher report. This allows the crawler to visit and download the most important pages first. In the proposed WBC, the rank of the website pages is calculated according to the frequency of updating the page, and its closeness to the main page. As both of these two factors determine the importance of webpages, an equal weight of 0.5 has been assigned to each one. The calculation details of these factors are summarized below:

a) Frequency of updating the pages: Initially, the frequency value of all pages is set to zero. Whenever a page is updated, its frequency is incremented. Then, the frequency values will be normalized to be within the range $\{0, 0.5\}$. The normalized frequency of page i (F_{n_i}) is computed as

$$F_{n_i} = \frac{1}{2} \left(\frac{F_i - F_{\min}}{F_{\max} - F_{\min}} \right) \quad \text{Eq.(3.1)}$$

where F_i is the frequency of updating the i^{th} page, F_{\min} and F_{\max} are, respectively, the smallest and the largest assigned frequency values.

b) Closeness of the page to the root (main page): The main page and the pages that are linked to it get the highest level value, i.e., 0.5. All other pages are categorized and get a discrete level value of 0.1, 0.2, 0.3, or 0.4, depending on their closeness to the main page. The number of categories N_{cat} is specified by applying the rule of thumb [63] on the *length of the website tree* and computed as

$$N_{cat} = \left\lceil \sqrt{\frac{length-1}{2}} \right\rceil \quad \text{Eq.(3.2)}$$

Finally, the rank of each page is calculated by adding its frequency and level values. It is worth mentioning that the watcher file saves and updates the frequency, the level and the rank values of each page in the report.

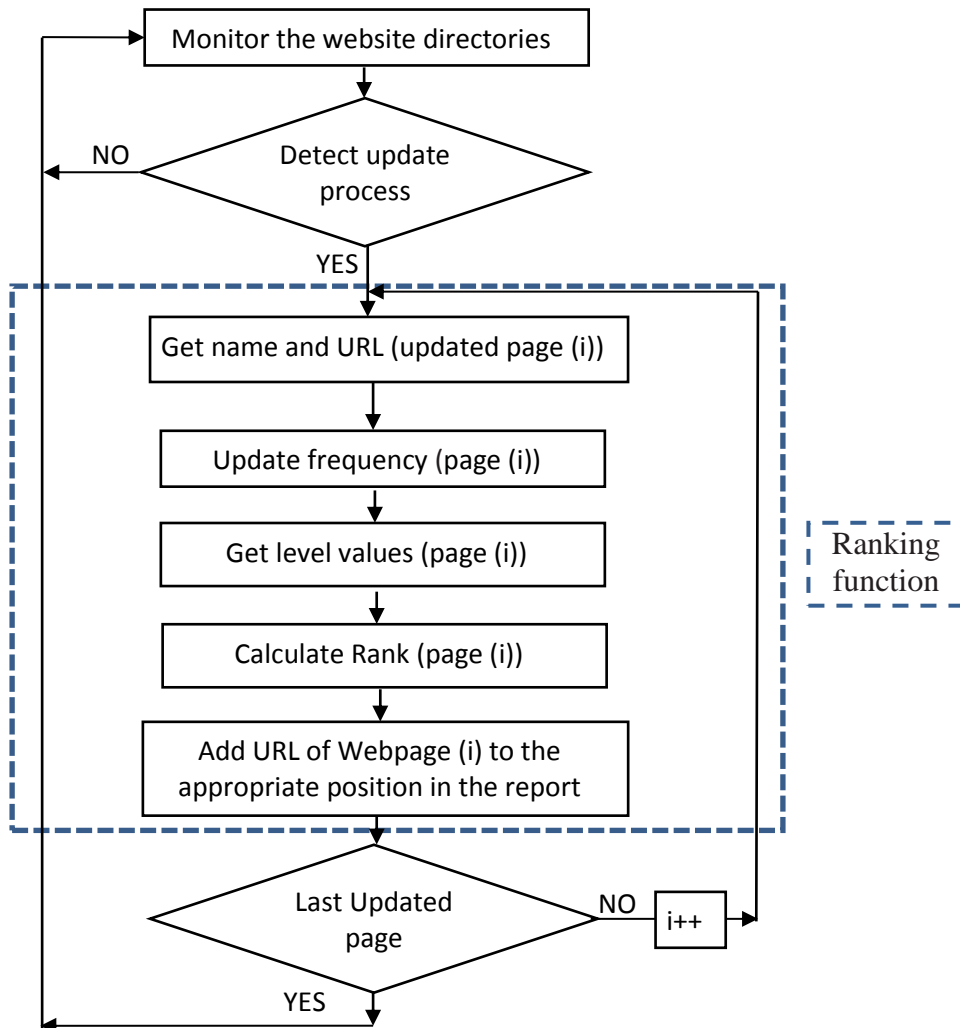


Figure 3.3: Flowchart of the developed watcher file for adding the updated static pages to the report.

3.4.1.2 Watcher File Setup

To run the watcher file on the web server, the administrator has to upload it to the main directory. Initially, the flag is re-set to zero, and this allows the WBC to visit the website. Then, if the website is processed by the WBC, its flag will be set to one. This prevents other WBC copies of visiting such website. In the developed WBC, two strategies that re-set the website flag have been implemented. Furthermore, the search engine administrator has the ability to select one of these re-set strategies. The details of these strategies are described below.

A. Event-Based- Re-set Strategy

In this strategy, the watcher file re-set the flag automatically when any of the website pages is updated. This increases the chance of re-visiting this website. On the other hand, visiting the same website many times may lead to decreasing the chance of visiting other lower ranked websites.

B. Time-Based- Re-set Strategy

In this strategy, the watcher file re-set the flag after a period of time which is assigned by the search engine administrator. The main advantage of this strategy is that all processed websites will be visited only once in a predefined period of time. It is important to note that if none of the website pages is updated during the predefined period of time, the flag will not be re-set, i.e. no pages to be processed by crawlers.

3.4.2 WBC-Server Design

To improve the performance of the developed WBC, the following schemes are used: First, multiple crawlers are run concurrently, where each crawler can run multiple threads. Second, the WBC work has been divided into five units: controller unit,

crawler unit, link extractor unit, AJAX unit, and link sorter unit, as shown in Figure 3.1. The functions of each unit are described below.

A. Controller unit: This unit is responsible for the following:

- I. Specify the number of working crawlers, and the associated threads.
- II. Cache the DNS tables: In the case of connecting a crawler to a website, it will contact with the DNS server to translate the website domain name into IP address [64]. As the DNS requests may require a large period of time, the controller unit of the proposed WBC is responsible for preparing and maintaining the WBC DNS caches. Hence, WBC crawler's frontiers contain the IPs instead of the domain names, and this will speed up the crawling performance.
- III. Distribute the URLs in the main frontier(s) between the running crawlers. In addition, the controller has the ability of updating the working crawler's frontiers.

B. Crawler unit: Multiple copies of this unit can run concurrently, where each copy is responsible for visiting the IPs in its frontier, reads the watcher report and downloads the updated pages in a directory named as "*LastDownloadedPages*". To avoid degrading the performance of any visited server, the developed WBC is designed in a way that it allows only one copy of the crawler unit to visit the server at the same time. This crawler unit has the ability of processing static and dynamic webpages as described below.

I. Processing static pages

The flowchart of the developed crawler unit for processing static webpages is shown in Figure 3.4. In the case of static pages, this unit gets the URLs of the

updated and the newly added pages from the watcher report. Then, it will check the Robot.txt file, which includes downloading permissions and specifies the files to be excluded by the crawler [65]. In the case that the crawler unit does not find the Robot.txt file, it will visit all updated and newly added website pages.

II. Processing dynamic pages

The process of indexing all dynamic pages requires a large period of time as the AJAX crawlers have to visit all webpages and search and trigger AJAX events to reach the dynamic information. The developed watcher file has been designed to detect and add the AJAX events including its triggering path to the watcher report. Hence, when the crawler visits the website, it downloads the dynamic pages and its report to be processed by the AJAX unit. This mechanism decreases the processing time and hence improves the crawling efficiency.

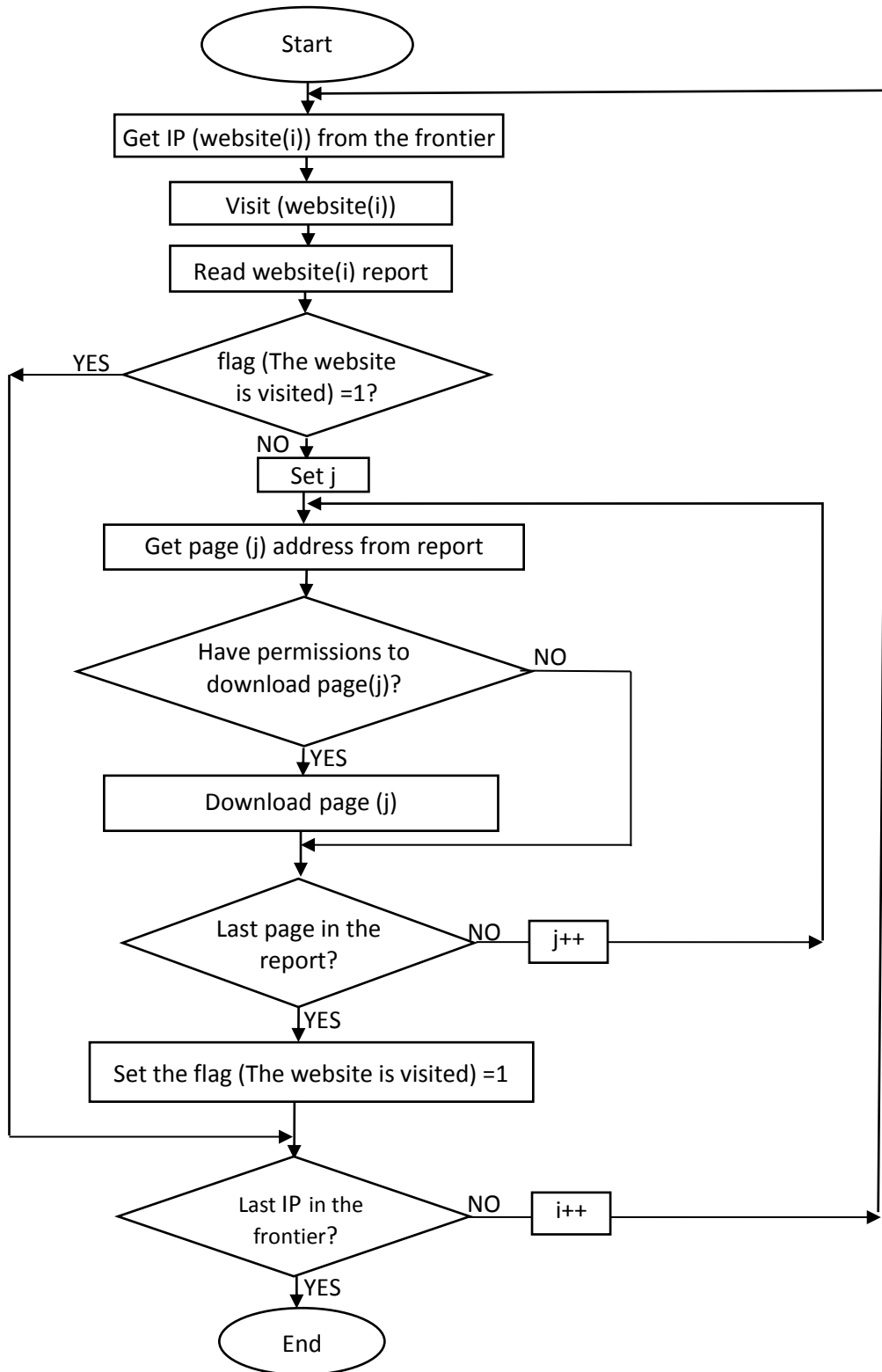


Figure 3.4: Flowchart of the developed crawler unit for processing static webpages.

C. Link extractor unit: This unit is responsible for getting the pages from the “*LastDownloadedPages*” directory, extracts and saves the pages URLs in a queue named as “*ExtractedURLsQueue*”. In this unit, the following are executed:

- I. Remove the processed pages from the “*LastDownloadedPages*” directory and save the static pages in the database. In addition, it moves the dynamic pages to a directory named as “*LastDownloadedAJAX Pages*” to be processed by the AJAX unit.
- II. In order to obey the webmaster restriction, the link extractor discards all URLs whose attribute “*rel*” set to “*nofollow*”.
- III. Add the URL of the website main page only to the “*ExtractedURLsQueue*”. This is because the watcher report provides the crawler the URLs of the updated pages. This is done by abstracting the main website URL for each of the extracted URLs. For example, all “*cmpe.emu.edu.tr*” website pages such as *cmpe.emu.edu.tr/FacultyMemberList.aspx* are abstracted to *cmpe.emu.edu.tr*. It is worth mentioning that most of the current crawler approaches extract all the URLs of the visited pages, and these URLs are visited and processed as well. This process consumes the system resources such as the memory, CPU cycles, etc. In addition, the crawler has to request a connection to a web server whenever it has a URL of a page hosted on it.
- IV. Removes the duplicated URLs from “*ExtractedURLsQueue*”.

D. AJAX unit: This unit is responsible for getting the dynamic pages from the “*LastDownloadedAjaxPages*” directory and triggering the events that have been reported by the watcher file instead of triggering all page’s events. Figure 3.5 shows the flowchart for processing the dynamic pages by the AJAX unit.

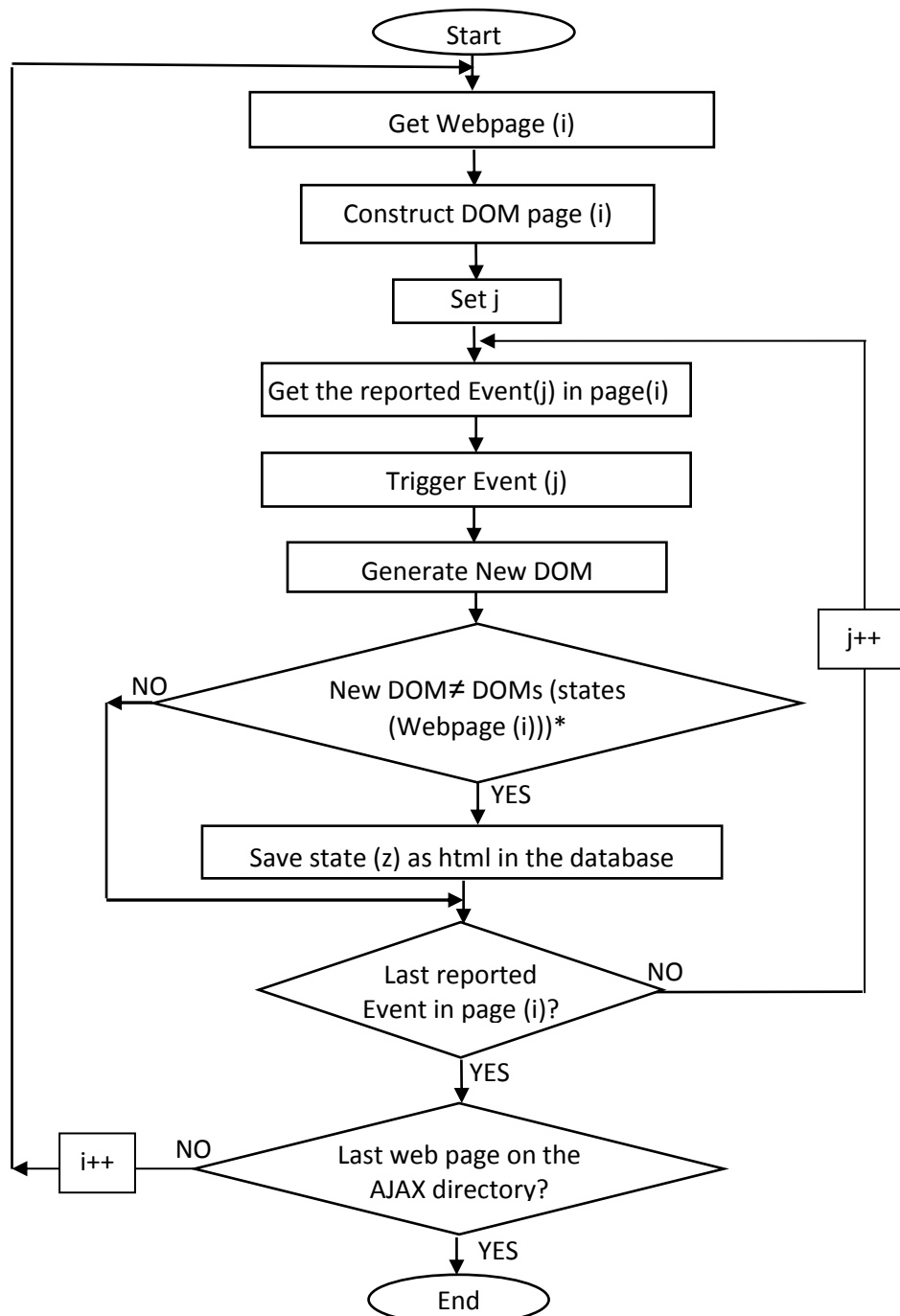


Figure 3.5: Flowchart for processing the dynamic pages by the AJAX unit. *This process is done according to Algorithm 3.1.

Based on Figure 3.5, to detect new states, the DOM of the generated state is compared with the DOMs of the previously founded states. This is done by applying the following algorithm:

```

1: Re-set flag=false; // The variable flag is set to true if the generated state is new.
   // NGD refers to the new generated DOM.
2: Remove the information of useless and irrelevant tags form the NGD
3: For i=1 to N do // N: Number of the previously founded states.
4:   For j=1 to M do // M: Number of elements (tags) in the generated state.
   // E is an element (tag) in DOM(I) and NGD
5:     If ( $E_{DOM(i)}^j$ .Contents !=  $E_{NGD}^j$ .Contents) then
6:       flag =true;
7:       Break;
8:     ELSE
9:       flag =false;
10:    End If
11:  End For
12:  If (flag==false) then //The NGD and DOM (i) are identical
13:    Exit;
14:  End If
15: End For
16: If (flag==true) then
17:   Set NGD as one of the page state
18:   Save the html of NGD in the database
19: End If

```

Algorithm 3.1: Detecting the AJAX page's distinct states.

It is worth mentioning that, webpages may contain useless and irrelevant information for crawling, such as advertisements, timestamps and counters, which are changed very frequently [66] and to insure that such information will not mislead the comparison process, the AJAX unit deletes these tags from the generated state, as mentioned in Algorithm 3.1, step 2. Finally, in order to obtain the dynamic content, the JavaScript engine V8 [67], and the embedded web browser Chromium [68], are used by this unit to parse the JavaScript code in a webpage, trigger its reported events, and constricting the new DOMs.

E. Link sorter unit: This unit is responsible for getting the URLs from the “*ExtractedURLsQueue*” and adds those URLs to the appropriate position in the main URLs frontier(s) based on the sorting process. In this work, we have adopted the back link count algorithm [69, 70] to sort the URLs. Moreover, the

link sorter unit has the ability of using a group of main frontier. In such case, each frontier has a rank depending on the importance and the rank of its URLs. Hence, the URLs of most important frontier(s) are visited faster and more frequently than the URLs of the lower ranked frontier(s).

3.4.3 WBC Complexity Analysis

As of February 2015, it has been mentioned in [49] that the total number of the published websites on the Internet was over 1.25 billion. Hence, visiting and processing all published websites is a challenging task. Most of the conventional crawlers have to visit and process all web pages in each website, and its complexity in terms of the number of required visits is $O(W*N)$, where W is the total number of published websites and N is the total number of web pages in each website. On the other hand, the complexity of WBC in its worst case is of the order of $O(W*U)$, and in its best case is of order $O(W)$, where U is the number of updated web pages. As $U \ll N$, the number of required visits of the WBC is much less than the number of required visits by the conventional crawlers.

3.4.4 WBC Properties

The developed WBC has the following properties:

1. Low cost and high performance: The watcher file increases the number of visited websites at almost no cost. The website administrator can get the watcher for free.
2. Parallel independent crawlers: Many independent crawlers can work in parallel with multiple threads.
3. The watcher file has the ability to perform its duties on the hosting servers, i.e., one copy of the watcher file can monitor a group of websites that are hosted on the same server.

4. The running crawlers can only read the watcher report and have no permission to control or contact the watcher file itself. This feature protects the websites servers and its data from any possibility of violation or attacked by spams softwares through using the watcher file.
5. Failure recovery: In the case that the controller unit did not receive information from any working crawler, the controller will consider that crawler as a dead one and the IPs in its frontier will be assigned to a new crawler.
6. Dynamicity of the assignment function: The controller unit has the ability to analyse previous work of the crawler and build statistic reports that helps in estimating the number of required crawlers and balancing the distribution of URLs. In addition, if any frontier contains a large number of IPs, the controller can run new crawlers and re-divides the IPs. Hence, the time required to visit all IPs is decreased.
7. By making use of the watcher report that provides the crawler the URLs of the updated pages, the “*ExtractedURLsQueue*” will contain only the main websites URL. This process decreases the number of URLs in the “*ExtractedURLsQueue*” and in the main frontier(s). This not only save the system resources but also significantly decreases the number of connection requests to each web server.
8. Flexibility of the assignment function: By making use of the flag, that has been introduced for solving the overlapping problem, the proposed WBC can work with any assignment function.
9. To avoid degrading the performance of the WBC server(s), if any of the directory or queue of the server units is found to be empty, the corresponding

unit will be set to inactive mode, and will be re-activated when new data is added to its corresponding path.

10. The watcher file has the ability to set the services of the ranking function to inactive mode during the server rush times, and re-activate these services later. This feature avoids degrading the performance of the website server.
11. Unlike, sitemap crawler, the proposed WBC perform all crawling processes in the sense that it detects the updated and the newly added pages automatically without any human explicit intervention or downloading the entire websites.

Chapter 4

THE PROPOSED RE-RANKING APPROACH

4.1 Introduction

Nowadays, re-ranking techniques are needed to improve the quality of the retrieved information. In general, re-ranking work based on re-order the original results of query, to show the most relevant files at the top of results list. In this chapter, an efficient re-ranking scheme is introduced.

4.2 Related Works

In [34-41, 71, 72], multiple content based re-ranking approaches have been developed to improve the quality of the retrieved multimedia files. In [34], it has been assumed that the images that are clicked in a response to a query are relevant ones, and the similar files will be ranked to the top of the new re-ranked list. However, in some cases, some checked images may appear to be irrelevant and in this case the approach of [34] will lead to get absolutely irrelevant files. In [35], an automatic re-ranking process, which works on integrating the files keywords and visual features, has been implemented for images only. In [41], a video re-ranking scheme that re-evaluates the rank of the video shots by the homogeneity and the nature of the video itself was proposed. The main disadvantage of this approach is that the process of estimating the rank will be done independently for all video shots in the database, and this will require a large processing time. In [71] and [72], other re-ranking algorithms have been introduced. These approaches are based on the

relationships between the files. The inclusive and exclusive relationships between semantic concepts are utilized to find the files relation in [71]. The inclusive relationship refers to the high co-occurrence relation between files, and the exclusive relationship refers to low or none co-occurrence relation between files. Then, the re-ranking of the retrieved results will be based on the average for all values of the impact weights between the attributes. In [72], a multiple pair-wise relationships between files were proposed. The set of the pair-wise features are used to capture various kinds of pair-wise relationships. Then, the extracted pair-wise relationships will be combined with a base ranking function to re-rank the original results. Although, the approaches of [71] and [72], can be used for multimedia files re-ranking and improves the percentage of relevant files. Those techniques have to find the relationship between a large numbers of objects and this will increases the required processing time.

In recent years, cross-media retrieval systems [35-41], where the type of the query example and the returned results can be different such as submitting an image of an object to retrieve its text description, were developed. However, finding the semantic correlation and the heterogeneous similarity of different multimedia modalities (cross-media) is still a challenging problem.

The main objective of the work presented in this chapter is to build a new re-ranking approach that can efficiently deal with all multimedia files. The proposed approach is based on the multimedia contents and some user specific actions such as download, copy a file or a part of a file or spending more than a number of seconds (N) in checking the file. The approach has the ability of accurately representing the

multimedia file using a group of descriptors that can be extracted concurrently through multiple threads. In addition, the weight of these descriptors is dynamically calculated and changed from a file to another based on the descriptors ability to distinguish between the files. Furthermore, unlike most of the conventional re-ranking approaches, the developed approach does not require any tuned parameters. Finally, the re-ranking process does not require any explicit user intervention, and it works is based on detecting some implicit user actions. Several experiments have been conducted and it has been observed that the developed re-ranking approach has the ability of showing the most relevant files to the top of the query results, and increases the percentage of retrieved relevant files.

4.3 The Proposed Re-ranking Structure

In the proposed re-ranking approach, the user actions will be detected and will be taken into account to improve the percentage of the retrieved relevant files. We believe that if a user performs one of the following actions on any of query results (files), it means that this file, which is referred in this work as *Target*, is related to the required ones:

1. Download the file.
2. Copy the file or a part of the file.
3. Spend more than a number of seconds (N) in checking the file, where N will be specified by the system administrator.

Then, the *Target* file will be analyzed and the query results will be reordered depending on their similarity with the *Target*. The structure of the proposed re-ranking approach is shown in Figure 4.1. The proposed approach contains offline operations like pre-processing and features extraction and online operations like

calculating files similarity and re-ordering the query results. In the following, the details of these operations are described.

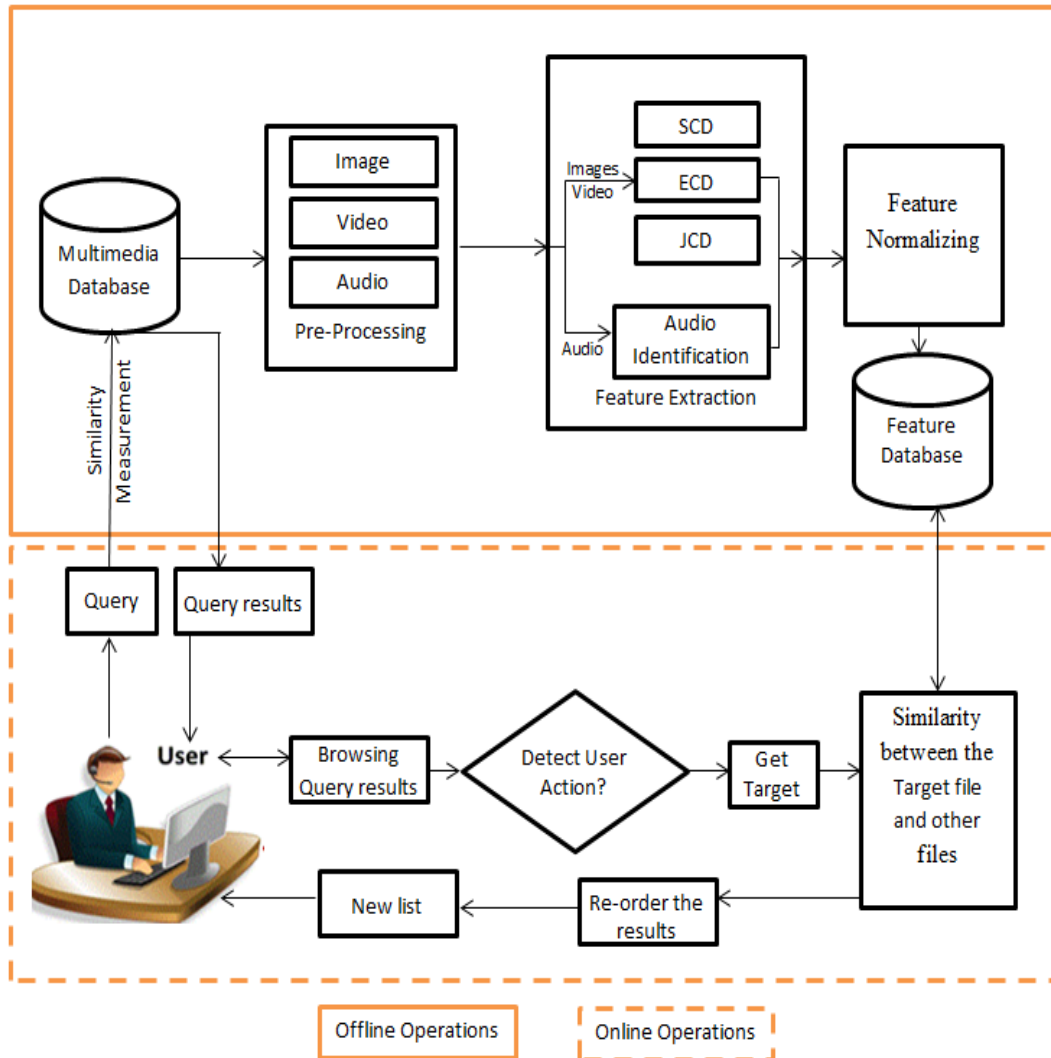


Figure 4.1: The structure of the proposed re-ranking approach.

4.3.1 Offline Operations

In the case of adding a new file to the database, the file will be pre-processed and its features will be extracted and saved. The details of these operations are explained below.

4.3.1.1 Pre-processing Operations

Nowadays, capturing and editing multimedia files is frequently used, and this may increase the noise in multimedia files, which has a strong influence on the quality of multimedia retrieval systems [73]. In the proposed approach, the noisy and the useless parts of multimedia files will be removed. The pre-processing operations for each multimedia type are performed using the most popular methods as described below.

I. Image files pre-processing operation

In general, image noise reduction methods are categorized as linear, nonlinear like median filter [75] and weighted median filter [75], and fuzzy methods [74, 75]. It has been found that linear filters may destroy image details, especially the edges. Fuzzy filters were found to provide promising result for image-processing tasks [74, 75]. However, it's processing time make it unsuitable for large image databases.

In this work, we have adapted the median filter which was found to be effective in keeping image detail at reasonable cost. It is worth mentioning that if the image contains text, the deskewing approach presented in [76] will be used to remove any skew.

II. Video files pre-processing operation

Video structure contains large number of consecutive frames combined with the audio information. Each video has redundant information, and it has been observed that videos include some meaningless frames, like totally black frames, totally white frames, faded frames, etc. [77]. This makes the process of selecting efficient features for representing video files difficult and time consuming. In this work, static video

summarization method [78, 79], which is found to be efficient and requires less time than dynamic method [78, 79], is used to filter and remove the irrelevant and redundant video frames. This is done by applying the following steps:

- 1) The meaningless frames will be detected and discarded by checking if the standard deviation of pixels in a frame is close to zero [77].
- 2) Assign a weight for each frame to represent the average of the frame similarity with other frames in the same video. The similarity between the frames will be calculated as

$$\text{Similarity}(\text{frame}(i), \text{frame}(j)) = \frac{N_s}{N_t}, \text{ for } i=1, \dots, N, j=1, \dots, N, i \neq j \quad \text{Eq.(4.1)}$$

where N is the total number of frames, N_s is the number of similar features in frame (i) and frame (j), and N_t is the number of features in the used descriptors, which is explained in details in the following section. Then, the weight of i^{th} frame is calculated as

$$\text{Weight}(\text{frame}(i)) = \frac{\sum_{j=1, j \neq i}^N \text{similarity}(\text{frame}(i), \text{frame}(j))}{N-1} \quad \text{Eq.(4.2)}$$

Finally, frames that have almost the same weight will be grouped and represented by one frame referred as key frame, while the other frames will be discarded.

III. Audio files pre-processing operation

Audio retrieval systems are more complicated than image and video systems, and the file features have to be selected carefully. This is because of many facts such as 1)

Audio formats (extensions) produce distinct file features, i.e., the features of two similar files but with different formats show that the files are different [80, 81], and

2) The combination of internal characteristics of the same audio format, such as bit rate, sampling rate, and number of channels, will indicate that two similar files are different [80, 81]. To solve such problems, the audio files will be decoded with Pulse Code Modulation format (PCM) [80, 81], and then, the sampling rate can be reduced by taking a discrete signal from the file's continuous signal [82]. This will lead to remove irrelevant information from the human perceptual point of view, and to focus on the important features of the signal. In this work, the audio files are sampled to 5512 Hz, which is considered to be safe and a required operation at the same time [80, 81].

4.3.1.2 Extract File Features

The features extraction is the process of extracting efficient representatives of multimedia files. One of the well-known challenges in content-based multimedia retrieval is selecting which features to represent the files. This is because selecting few features may not be sufficient to characterize the multimedia data, while selecting large number will make the system complex and time consuming. In the following, the feature extraction process for image, video and audio files is described:

I. Image feature extraction process

To represent the images accurately, we have selected the following descriptors:

- 1) Scalable Color Descriptor (SCD): This descriptor describes the color distribution in an image by obtaining the color histogram, which can provide the global color features when it is measured over the entire image [83].

- 2) Edge Histogram Descriptor (EHD): This descriptor describes the edges distribution in an image, which are important features to represent the images contents [84]. In addition, EHD descriptor helps in finding the semantic meaning of the image contents.
- 3) The Joint Composite Descriptor (JCD): This descriptor is a combination of the color and edge directivity descriptor (CEDD) [85], and the fuzzy color and texture histogram descriptor (FCTH) [86]. These two descriptors combine color and texture information in order to describe the file visual contents. The JCD was introduced in [87] based on the fact that the CEDD was found to be better for some multimedia files while the FCTH was found better for some others. In addition, it has been found in [87] that the JCD demonstrates high success rates in image databases.

In this work, the above descriptors have been utilized to ensure the quality of the extracted features. Each image file is represented in the database by three vectors containing the descriptors output as follows:

$$F_{dscr1} = [f_1; f_2; \dots; f_N]^T, F_{dscr2} = [f_1; f_2; \dots; f_M]^T, \text{ and } F_{dscr3} = [f_1; f_2; \dots; f_L]^T,$$

where $dscr1$, $dscr2$ and $dscr3$ represent the SCD, EHD, and JCD, respectively, and the values of N , M and L are 255 [83], 80 [84], and 168 [87], respectively.

II. Video feature extraction process

In order to avoid increasing the computational cost of processing large videos databases, file fingerprinting, which is a content-based compact signature that summarizes the characteristics of the file and can distinguish between the files, has

been used [88]. In addition, the video fingerprint will be represented by three vectors only, where each vector contains the average of one of the used descriptor values for all video key frames. In this case, the video Key frames can be represented as

$$[K_1; K_2 \dots ; K_{KF}]^T,$$

where KF is the number of Key frames which varies from a video to another, and the video features are represented as

$$F_{dscr1} = [\bar{f}_1; \bar{f}_2; \dots; \bar{f}_N]^T, F_{dscr2} = [\bar{f}_1; \bar{f}_2; \dots; \bar{f}_M]^T, \text{ and } F_{dscr3} = [\bar{f}_1; \bar{f}_2; \dots; \bar{f}_L]^T,$$

where the value of \bar{f}_i for the $dscr_j$ is calculated by finding the average of f_i obtained by $dscr_j$ for all the video key frames, where $j=1, 2, 3$, and $i = 0-255$ for SCD [83], 0-80 for EHD [84], and 0-168 for JCD [87].

III. Audio feature extraction process

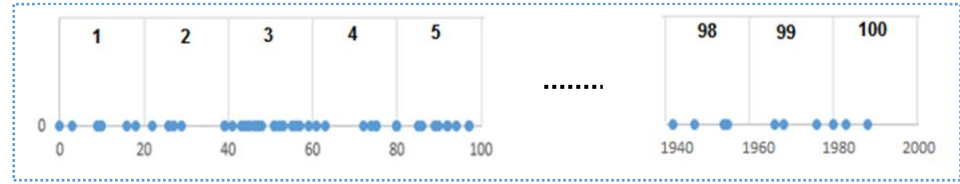
The features of audio files can be classified into local features, which represent the characteristic of a specific part of the audio file, and global features, which represent the characteristic of the entire file [88]. These two types may suffer from the problems discussed in the pre-processing section, i.e., the features of two similar files may show that these files are different. Nowadays, audio fingerprinting, also known as audio identification, is used to summarize an audio file by finding its unique characteristics. In this work, the fingerprints of the audio files will be constructed through the following steps:

- 1) Building the spectrogram: After preprocessing the signal with 5512 Hz PCM, the audio spectrogram, which shows how the spectral density of a signal

varies in time, will be constructed [80, 81]. In this work, the file spectrogram is built by dividing the signal into overlapping frames, and then each frame is passed through a Fast Fourier Transform [80, 81] to get the spectral density variation in time domain.

- 2) Band filtering: Generally human ears can recognize the frequencies in the range 20 Hz-2000Hz. In this work, the spectrogram is filtered to obtain this range.
- 3) Reducing the length of the fingerprint: The audio file is represented by a group of frames, which produce a large number of feature points, i.e., the fingerprint of each audio file is represented by 8192 points. One important aim of the developed approach is that it can deal with large databases efficiently. This can be achieved by reducing the file fingerprint. To achieve this, we have proposed a method that can reduce the fingerprint length. This method is based on chain code algorithm [82]. The details of the proposed method are summarized below.
 - i. Represent the file fingerprint in a two dimensional chart. In this chart, each fingerprint point is represented by its value on the x axes and a zero on the y axes. Based on the experimental work, it has been found that assigning a zero on the y axes for all points not only decreasing the processing time, but also, the produced fingerprint represents the unique properties of each audio file.
 - ii. Divide the chart into group of blocks, where each block may contain different number of points. In this step, the fingerprint chart will be divided into 100 blocks. This number was selected as it was shown in [80, 81] that it can give the unique properties of the audio file. For

instance, the following diagram shows a snapshot of an audio feature points.



- iii. Represent each block in the audio fingerprint by a value. This value is computed by applying the following:
- a) Eliminate all points allocated at the borders of the blocks.
 - b) Represent any empty blocks (if any) by a zero.
 - c) Represent other blocks by the average of the points allocated inside it.

After applying a-c steps, the produced audio fingerprint is represented as

$$F = [f_1; f_2; \dots; f_Z]^T$$

where $Z = 100$, and f_i is the average of the i^{th} square points or zero. Based on the experimental work, we have noticed that in some cases, the fingerprint may not reflect effectively the similarity between some files especially those that have different lengths. To overcome this problem, each audio file will be virtually divided into approximately three equal parts, and then the fingerprint for each part is obtained by applying 1-3 steps. It is worth mentioning that the three fingerprints of a file are found concurrently using multiple threads. Finally, each audio file is represented by three vectors as:

$$F_{\text{Sign1}} = [f_1; f_2; \dots; F_Z]^T, F_{\text{Sign2}} = [f_1; f_2; \dots; F_Z]^T \text{ and } F_{\text{Sign3}} = [f_1; f_2; \dots; F_Z]^T,$$

where $Sign1$, $Sign2$ and $Sign3$ represent the fingerprint of the first, the second and the third part of the file, respectively.

Finally, the extracted features will be normalized and adjusted to be in the same range.

4.3.2 Online Operations

In this section the user behavior will be monitored, and when the user performs one of the following actions: download, copy, or spending more than a number of seconds (N) with a Target file, the query results will be re-ranked by applying the following steps:

- 1) Calculate the distance between the Target file (T) and the other files (Y) by using the Euclidean formula [83]

$$D_{dscr_j}(T, Y) = \sqrt{\sum_{i=1}^n |f_i(T) - f_i(Y)|^2}, \quad \text{for } j=1, 2, 3 \quad \text{Eq.(4.3)}$$

where the descriptors for video/image files are SCD, EHD, and JCD, while they are the file's three fingerprints for the audio files.

- 2) Assign a weight for each descriptor that specifies its influence on the file rank. This is based on the similarity of the descriptor vector of the Y and T files. In the case of images and videos, the descriptor that has the ability to distinguish between the files will be assigned a higher weight than the others, while the opposite is applied in the case of audios. This is because the proposed audio signature summarizes an audio file by finding its unique characteristics, and therefore, when the distance of the signature values for Y and T files is low means that these files are related to each other. In this work, the weights sum

of the used descriptors is assumed to be 1, i.e., $\alpha + \beta + \gamma = 1$, where α , β and γ are the weights of the used descriptors, which are computed for the images/videos as

$$\alpha = \frac{D_{\text{dscr1}}(T,Y)}{T_D(T,Y)} \quad \text{Eq.(4.4)}$$

$$\beta = \frac{D_{\text{dscr2}}(T,Y)}{T_D(T,Y)} \quad \text{Eq.(4.5)}$$

$$\gamma = \frac{D_{\text{dscr3}}(T,Y)}{T_D(T,Y)} \quad \text{Eq.(4.6)}$$

where T_D is the total distance between T and Y files computed as

$$T_D(T, Y) = D_{\text{dscr1}}(T, Y) + D_{\text{dscr2}}(T, Y) + D_{\text{dscr3}}(T, Y) \quad \text{Eq.(4.7)}$$

while for the audios, α , β and γ are computed as

$$\alpha = \frac{\bar{\alpha}}{\bar{\alpha} + \bar{\beta} + \bar{\gamma}} \quad \text{Eq.(4.8)}$$

$$\beta = \frac{\bar{\beta}}{\bar{\alpha} + \bar{\beta} + \bar{\gamma}} \quad \text{Eq.(4.9)}$$

$$\gamma = \frac{\bar{\gamma}}{\bar{\alpha} + \bar{\beta} + \bar{\gamma}} \quad \text{Eq.(4.10)}$$

with

$$\bar{\alpha} = \frac{T_D(T,Y)}{D_{\text{dscr1}}(T,Y)} \quad \text{Eq.(4.11)}$$

$$\bar{\beta} = \frac{T_D(T,Y)}{D_{\text{dscr2}}(T,Y)} \quad \text{Eq.(4.12)}$$

$$\bar{\gamma} = \frac{T_D(T,Y)}{D_{\text{dscr3}}(T,Y)} \quad \text{Eq.(4.13)}$$

3) The rank of the file (Y) is computed as

$$R(Y) = \alpha D_{\text{dscr1}}(T, Y) + \beta D_{\text{dscr2}}(T, Y) + \gamma D_{\text{dscr3}}(T, Y) \quad \text{Eq.(4.14)}$$

4) Finally, in order to ensure the rank value to be in the range $[0, 1]$, the rank of file (Y) need to be normalized. The file rank normalized is donated as N_R^Y and computed as

$$N_R^Y = \frac{R(Y) - R_{\min}}{R_{\max} - R_{\min}} \quad \text{Eq.(4.15)}$$

where R_{\min} and R_{\max} are, respectively, the smallest and the largest assigned rank value.

Chapter 5

THE PROPOSED ELIMINATION APPROACH FOR DUPLICATED MULTIMEDIA FILES

5.1 Introduction

Recently, it has been found that around 40% of published webpages and its contents are duplicated [11-14]. In the following, the main problems caused by the huge amount of duplicated files are described:

1. The chance of getting the required files is reduced, and the order of the relevant files is negatively affected. It is well known that most of the users check only the first twenty results of the query [91], and because of this, the user may find few relevant files.
2. Increase both the search engine process time and the user time for examining the results.

In last few years, several approaches have been introduced to eliminate the duplicated multimedia files [11]-[14], [92-96]. In [12, 13], an approach for eliminating the repetition in image search engines has been developed. This approach creates an image database and calculates the Hash values [16, 97] for each image. Then, it compares the Hash values to find the repetitions and marks only one copy of the repeated image files. In [94], a framework for eliminating near-duplicate videos on social web has been proposed. Near-duplicate video means that the contents of the

videos are identical but the videos are different in one or more of their characters, i.e., formats, caption, logo, time duration, etc. The framework of [94] is based on combining the contextual information with the contents of the files to find and eliminate the near-duplicate videos from the top rank list. The comparison process in this approach is done on each query, and this will increase the query time. In [95], the contextual information such as time duration, number of views, and thumbnail images have been combined with the file content information, such as colour and local points, to find and eliminate the near-duplicate videos. Finally, an audio approach that detects the duplicated files has been introduced in [80, 81]. This approach is based on the audio fingerprint technique [80, 81] that produces exactly the same value for each file and its duplicated copies.

The main objective of the work presented in this chapter is to improve the efficiency of multimedia search engines by eliminating the duplicated files. For this purpose, we have developed an elimination scheme, which can deal with any type of multimedia files like image, video and audio. In addition, the performance of the proposed elimination scheme is improved by parallel processing. Moreover, unlike recently developed approaches [94, 96], the proposed elimination scheme performs the comparison process offline and only once, i.e., during the creation and the updating process of the database(s).

5.2 The Proposed Elimination Structure

The scheme developed in this work uses the Hash [16, 97] and the feature extraction [17] techniques for performing the comparison process. The Hash algorithms will be used for image files, and the low level features are extracted for the video/audio files. As will be shown in the experimental work, the performance of the MD5 algorithm

for dealing with images outperforms other popular techniques [13]. Although, MD5 algorithm has the capability of dealing with video and audio, however, we have observed that this algorithm was not able to detect duplicated video and audio files that have different formats, i.e., extensions. Therefore, the low-level feature extraction [17] has been adapted in the video and audio comparison process. The algorithm of eliminating the duplication of the multimedia files during creating and/or adding new files to multimedia database(s) is summarized below, and its flowchart is shown in Figure 5.1.

```

// N= Number of new files which will be added to the database
1: For i=1 to N do
2:   Get the multimedia file (i)
3:   Set Flag (new file (i)) =1.
//Create the file properties (hash value using MD5 or extract low level features using MPEG-7).
4:   If (Type of (new file (i)) =Image)
5:     Properties (new file (i)) =MD5 (new file (i))
6:   Else
7:     Properties (new file (i)) = extract the features (new file(i))
8:   End If
9:   For j=1 to Y do //Y=Number of files in the database
10:    If properties (newfile(i))= properties(file(j)) then
11:      Re-set Flag (new file (i)) = 0
12:    Exit
13:    End If
14:  End for
15:  Save new file (i), Flag (new file (i)) and Properties (new file (i)) in the database.
16: End for

```

Algorithm 5.1: Eliminating the duplication of the multimedia files during creating and/or adding new files to multimedia database(s).

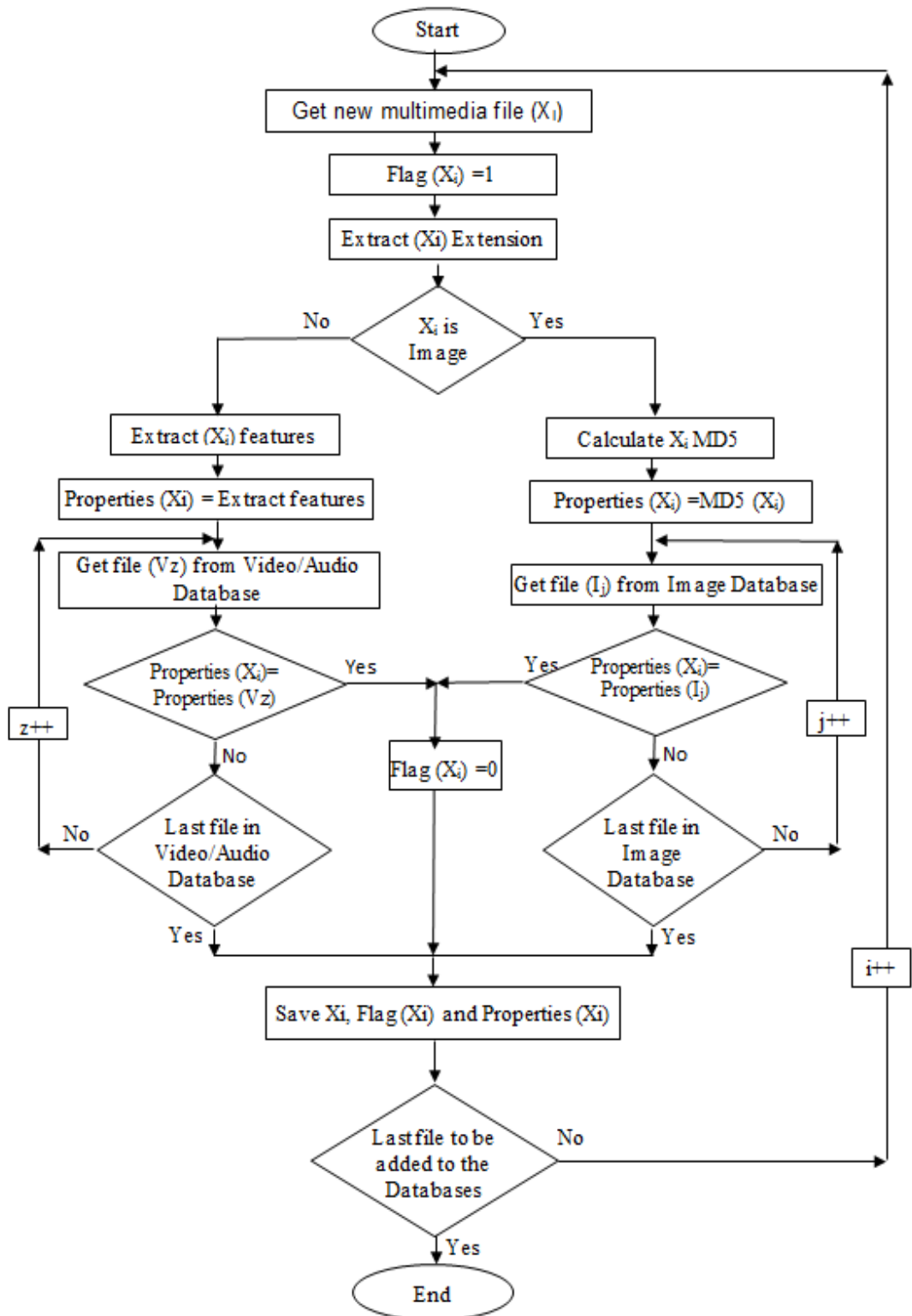


Figure 5.1: The flowchart of eliminating the duplication of the multimedia files during creating and/or adding new files to multimedia database(s).

The proposed elimination scheme has the advantage of working with any existing databases. In addition, it can be used for creating the databases itself. It is important to note that in the proposed elimination scheme, the websites ranking positions is kept unchanged, and the repeated files are not physically deleted from the database during the process of detecting duplicated files. Instead, a flag field is added to the multimedia database table(s). This flag is set to 1 for a file and will be set to 0 for its duplicated ones (if any). In addition, when processing a query, the system will only list files whose flag value is set to 1. Unlike most of the duplicated elimination approaches, in the proposed approach, the files properties are extracted once and stored in the database(s) during its creation rather than re-extract the properties repeatedly.

5.2.1 Hash Algorithms and Feature Extraction Techniques

5.2.1.1 Hash Algorithms

Hash algorithms are cryptography functions that take any information as input and convert it to a numeric code. The output of these algorithms is unique for each file, and it is like a fingerprint. By using Hash algorithms, files can be compared in fewer amounts of data. For instance, to compare two images only 16 bytes are used, rather than comparing all images pixels, i.e., Bit-wise comparison [16, 97]. In this work, a special Hash algorithm called Message-Digest 5 (MD5) [16] is used to find duplicated image files. MD5 Hash algorithm was chosen because it has the following advantages over other hashing algorithms:

- a) The size of the input file can be infinite.
- b) The output of MD5 is small in size (16 bytes) as compared to other Hash algorithms like SHA256 (32 bytes) and SHA512 (64 bytes), and hence, the comparison process using Md5 is less than the other Hash algorithms [13].

The steps of producing the MD5 hashing value [16] are briefly summarized below:

- a) Set the input: In this step, n -bit message is submitted as an MD5 input, where $n \geq 0$. In the case of image, the MD5 input will be an array of bytes derived from the image.
- b) Append padding bits: The message is extended so that its length in bits equals to $448 \bmod 512$. To do so, a single “1” bit is appended to the message, and then “0” bits are appended so that the length in bits equals $448 \bmod 512$.
- c) Append length: A 64-bit representation of the message length is appended to the result of step b. After applying step b and c, the length of the resulting message will be a multiple of 512 bits.
- d) Initialize MD buffer: Four buffers, which are known as A, B, C, and D are used to compute the message digest. Each of these buffers is a 32-bit register and initialized to a specific hexadecimal values.
- e) Process message in 512-bit blocks: In this step, four rounds of operations, where each round has 16 operations, are performed on each 512-bit block of the message and the four buffers, and the contents of the four buffers will be updated after each round.
- f) Output: In this step, the message signature is produced by combining the contents of A, B, C, D buffers starts with the low-order byte of A buffer, and ends with the high-order byte of D buffer.

5.2.1.2 Feature Extraction

Feature extraction refers to the process of getting a set of features (useful information) from the input file. In the proposed elimination scheme, we have extracted low level information from the multimedia files themselves by using Multimedia Content Description Interface feature (MPEG-7) [17]. In this manner,

the following video/audio file properties will be extracted: format, caption, logo, and time duration. In order to improve the performance of the proposed elimination scheme for detecting all the duplicated video and audio, the following properties will be extracted as well: In the case of audio, the file size, the bite rate, the sampling rate, and the number of channels. In the case of video, the features of the following descriptors will be extracted:

- a) Color layout descriptor: This descriptor describes the spatial distribution of color in the input file, which can be obtained by applying the discrete cosine transformation [98] on the local representative colors in Y or Cb or Cr color space [98].
- b) Texture descriptor [99]: This descriptor describes patterns such as directionality and regularity in video's frames. In order to describe the frame's texture, energy and energy deviation values are extracted.
- c) Contour-based descriptor [100]: This descriptor describes objects based on detecting its contours (borders).

5.2.2 Mechanisms of Multimedia Files Comparison Process

In the proposed elimination scheme, the Hash value will be computed for image files, and the comparison is performed by comparing the files hashing value (16 Bytes). In addition, low level features are extracted for the video and audio files, and the comparison process is performed by comparing the files extracted properties. The result of comparing video and audio properties can be categorized into three cases:

1. All the files properties have the same values. This indicates that the files are duplicated. Hence, the flag of the new file is set to 1, and the comparison process is stopped.

2. All the files properties are different. This indicates that the files are not duplicated, and the comparison process for these two files will be stopped, and the new file will be compared with the next one in the database.
3. Some files properties have the same values. This indicates that the files may or may not be duplicated. In the case of videos, the files descriptors values will be compared using Euclidean distance formula to find duplicated files. In the case of audios, the files signals decoded with Pulse Code Modulation format (PCM) will also be compared using Euclidean distance formula.

5.2.3 Parallel Implementation of the Proposed Elimination Scheme

The efficiency of the proposed elimination scheme can be increased by using parallel processing. In the case of having one database, the number of multimedia files is divided evenly between the running parallel processes. However, in the case of having more than one database, which is the situation of search engines, multiple processes will be run for each database. The server administrator can decide on the number of running processes depending on the total number of multimedia files in the database(s). It is important to note that in the case of the sequential elimination, each file of the database file is compared with the other files, and the comparison process stops as soon as a duplicated file is found. In the case of parallel elimination, the database files are divided evenly between the running processes, and then, each of the process will compare the files in its part with the other files. Similar to the sequential elimination scheme, the comparison process stops as soon as a duplicated file is found.

Chapter 6

THE ENHANCED QUERY BY EXAMPLE

6.1 Introduction

Query by Example (QBE) is a powerful search engine that allows the user to search for files based on an example [10]. QBE necessitates the user to upload a file, i.e., example file, to find the most relevant ones. In recent years, several developments in this field have been introduced [101-110]. In [101-106], Query by Sketch (QBS) was introduced to improve the performance of QBE. In QBS, the user will have more options, like using drawing tools, to describe what required exactly. In [106], a new content based image retrieval system has been proposed to improve QBS and QBE by using intelligent user interface agents, which is a combination of the dynamic user interface and the artificial intelligence techniques [106]. This system has four different types of user interfaces to perform image queries: keyword searching, category browsing, QBE and QBS. In [107], a video retrieval method based on QBE has been introduced. This approach is capable of filtering irrelevant video shots and selects example shots to perform the query without user supervision. On the other hand, its query processing may take a large time, especially for large duration videos [107]. In [108], a novel method to represent video data by developing an optical flow tensor (OFT), which is used for estimating the motion in the video was presented. This method incorporates Hidden Markov Models (HMMs) [108] to capture the implicit statistical correlations among video shots. In this method, the query video

shots will not be compared with all other shots recorded in the database. The precision of this method, however, does not exceed 70% [108]. In [109], a graph transformation and a matching scheme to find similar contents of a short video query within a long video sequence was presented. It has been shown that this method allows finding the smaller set of video candidates related to the query. In addition, this scheme can deal with videos whose shots have been reordered. Although this feature can increase the number of candidate videos, it can lead to obtain files which have no relation with the required ones. In [110], an audio QBE approach that estimates the similarity of the uploaded file and database files by calculating the distance using the probability density functions [110] was introduced. This approach may require large processing time as it finds the probability density functions for the features of each frame.

Although the above approaches have improved QBE performance, existing QBE still suffers from the following two problems:

- 1) Up to date, the efficiency of the QBE for video and audio files is much lower than QBE for image files [101-110].
- 2) The QBE query processing time is relatively large as compared with other search engine mechanisms like text based engines.

The main objective of the work presented in this chapter is to improve the overall performance of QBE for handling all multimedia types such as: image, video and audio. This is achieved through the following: First, a dynamic structure of clustering is used for creating and updating multimedia databases. Second, parallel implementation of the QBE has been implemented. Finally, the enhanced QBE has

the ability of eliminating duplicated multimedia files from the queries results by using the elimination scheme explained in Chapter 5. From the experiments conducted in Chapter 7, Section 7.4, it has been observed that the enhanced QBE supports all multimedia types, increases the percentage of relevant files and decreases the query processing time.

6.2 The Enhanced QBE Structure

The structure of the enhanced QBE is shown in Figure 6.1. The enhanced QBE consists of the following three main steps:

- 1) Collecting the data, which is done in most cases by using web crawlers.
- 2) Creating / updating the database.
- 3) Processing the query.

In this chapter, the QBE is enhanced by improving the performance of steps 2 and 3 by making use of clustering and parallel implementation techniques as described below.

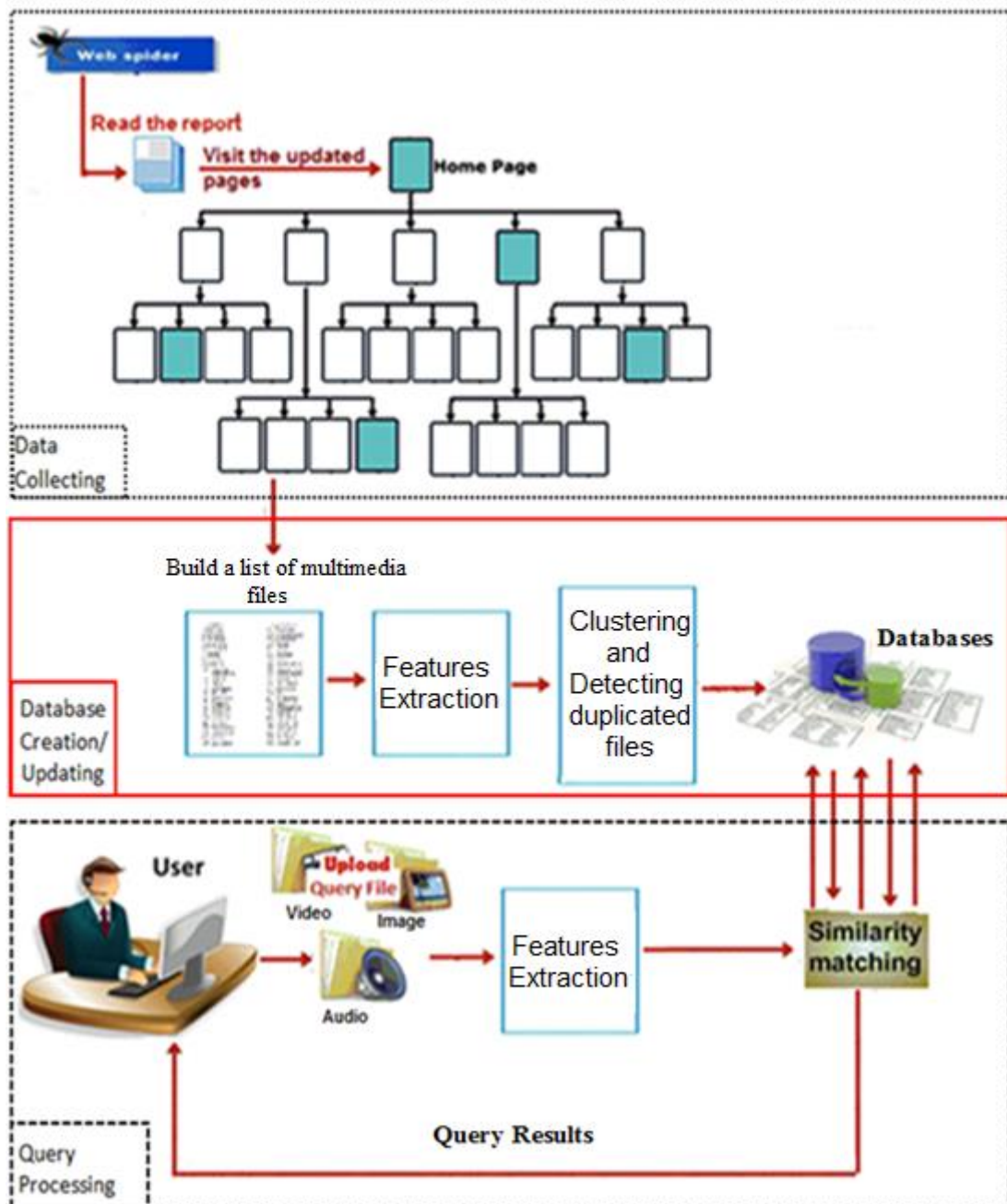


Figure 6.1: The structure of the enhanced QBE.

6.2.1 Clustering Techniques

Clustering [111, 112] is a technique that can be used to categorize multimedia files automatically in order to speed up the process of finding the queries relevant files. In general, clustering techniques can be categorized into two categories: hard, where each object belongs to a single cluster only, and soft, where each object can belong to

several clusters. We have noticed that the hard clustering outperforms the soft clustering when they are used with the enhanced QBE. It is important to note that, in the enhanced QBE, the clustering techniques will be used during the creation of the databases. The steps for creating and/or adding files to the database are shown in Figure 6.2 and summarized as follows:

- 1) Input: Get new multimedia file.
- 2) Pre-processing: This step involves removing unwanted and useless part of a file. The details of the pre-processing were explained in the re-ranking technique introduced in Chapter 4, Section 4.3.1.1.
- 3) Features extraction: This step is designed in order to allow QBE to deal with all multimedia types. The details of the feature extraction were explained in the re-ranking technique introduced in Chapter 4, Section 4.3.1.2.
- 4) Features normalization: The values of the extracted features are normalized and adjusted to be in the same range.
- 5) Multimedia clustering: In this step, the following clustering algorithms are used: k-means [113], subtractive [114], spectral [115], hierarchical [116], neural network [117]. Although the enhanced QBE was designed to work with these clustering algorithms, it has the ability to work with any other clustering algorithm.

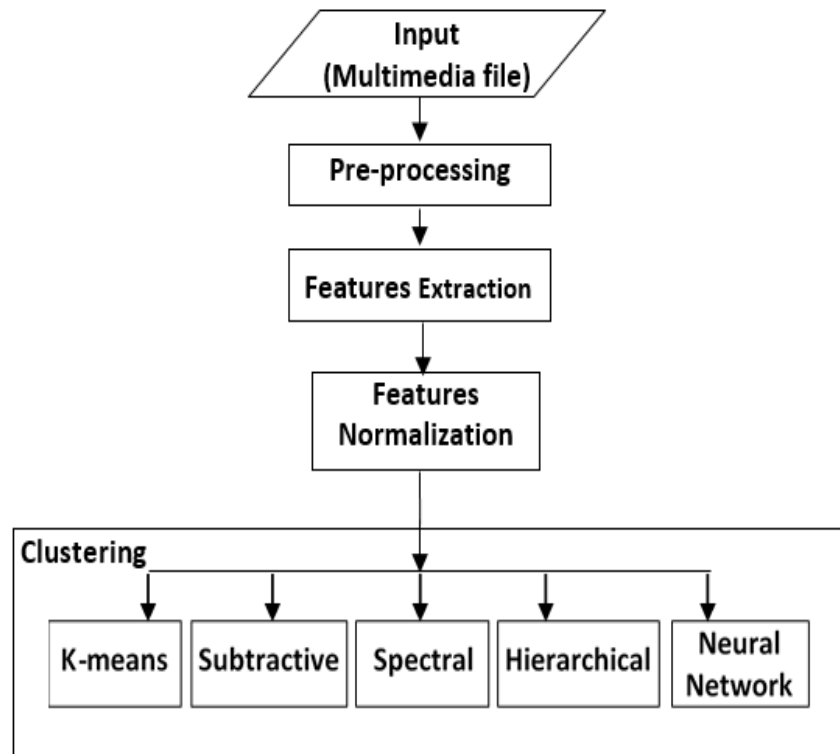


Figure 6.2: Steps of creating and/or updating the database(s).

To enhance the clustering performance and to eliminate the danger of specifying the number of clusters, which is known to be one of the clustering problems, we have decided to automatically specify the number of clusters. This is done based on the percentage of similarity between the files. The files will be clustered by using the following two steps:

- 1) Specify number of clusters: Initially, a cluster is created and one of the new files is defined as the cluster leader. If the similarity of next file with the leader of the previously created cluster(s) is under a threshold value, a new cluster will be created and that file will be defined as its leader. It is worth mentioning that this process will be done only once, i.e., when the database is created. The flowchart of this step can be seen in Figure 6.3.

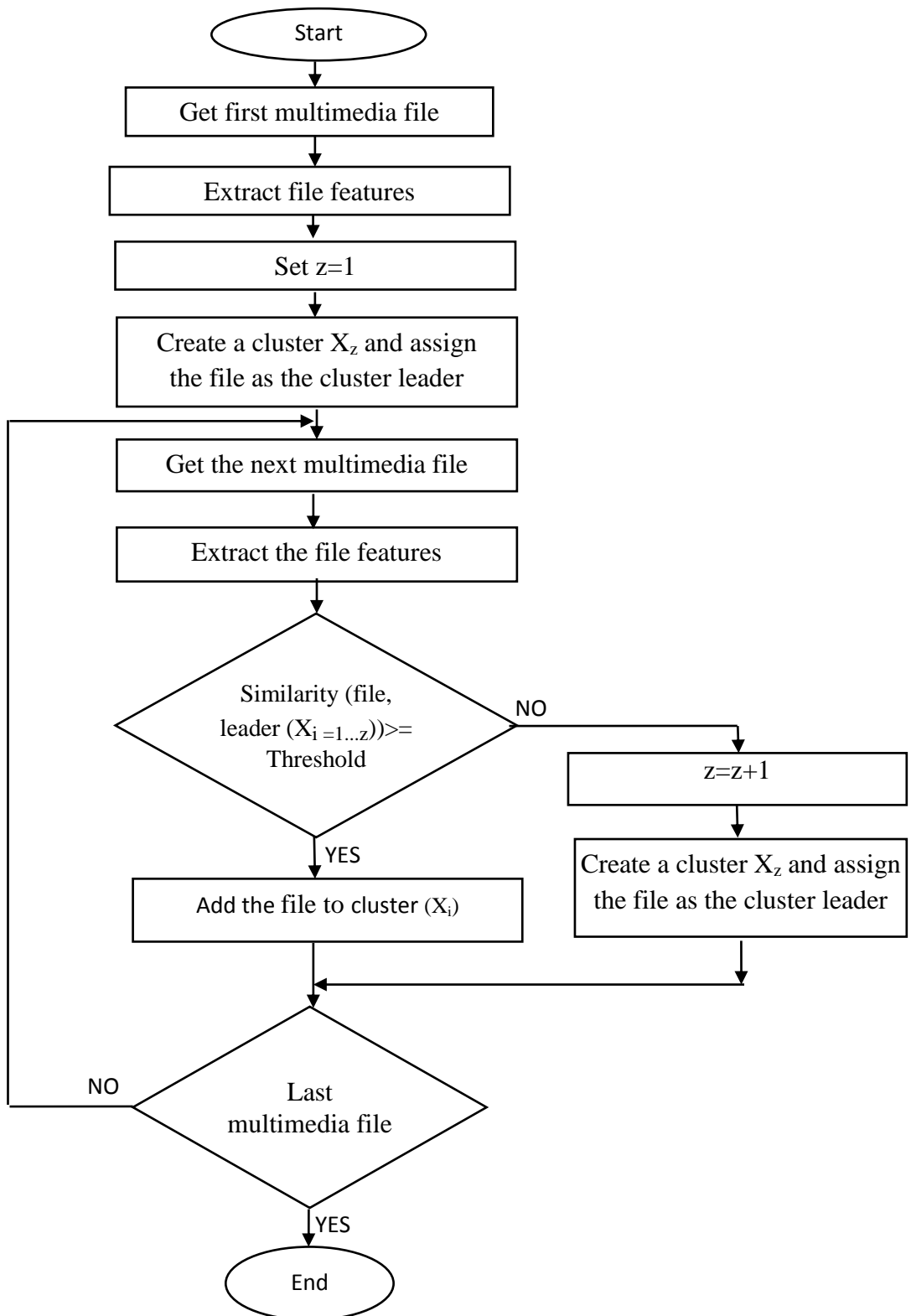


Figure 6.2: The flowchart to define the number of clusters, where z is number of clusters.

2) **Multimedia clustering:** In clustering all multimedia files, we used an ensemble clustering system, which is based on static classifier selection [118], and built by finding the best combination of the above mentioned clustering algorithms. From the experimental work mentioned in Chapter 7, Section 7.4, we have found that the best performance was obtained by using the combination of k-means, subtractive and spectral. It is important to note the following: First, the leader of each cluster created in this step is represented by the centroid of the cluster's files. Second, the majority voting method [118], which is widely used in clustering ensemble because of its simplicity, robustness and stability, was used in the clustering process. Third, during the process of the file clustering, if each member of the ensemble votes for different cluster, then the file will be saved in a new cluster.

6.2.2 Parallel Implementation of the Proposed QBE

In order to implement the enhanced QBE in parallel, it has been divided into the following two parts:

Part I: This part includes the main webpage, which will be used by the user to upload the multimedia file and to browse the query results.

Part II: This part is not shown to the user and implemented in parallel by running multiple threads. The aim of this part is to compare the features of the uploaded file and the leaders of clusters in the database(s).

It is important to note that in the case of having one database, the number of existing clusters will be divided equally between the running threads. However, in the case of having more than one database, multiple threads will run for each database. The flowchart of the proposed parallel implementation for one database is shown in

Figure 6.4. In the case of multiple databases, a copy of this implementation is used for each database.

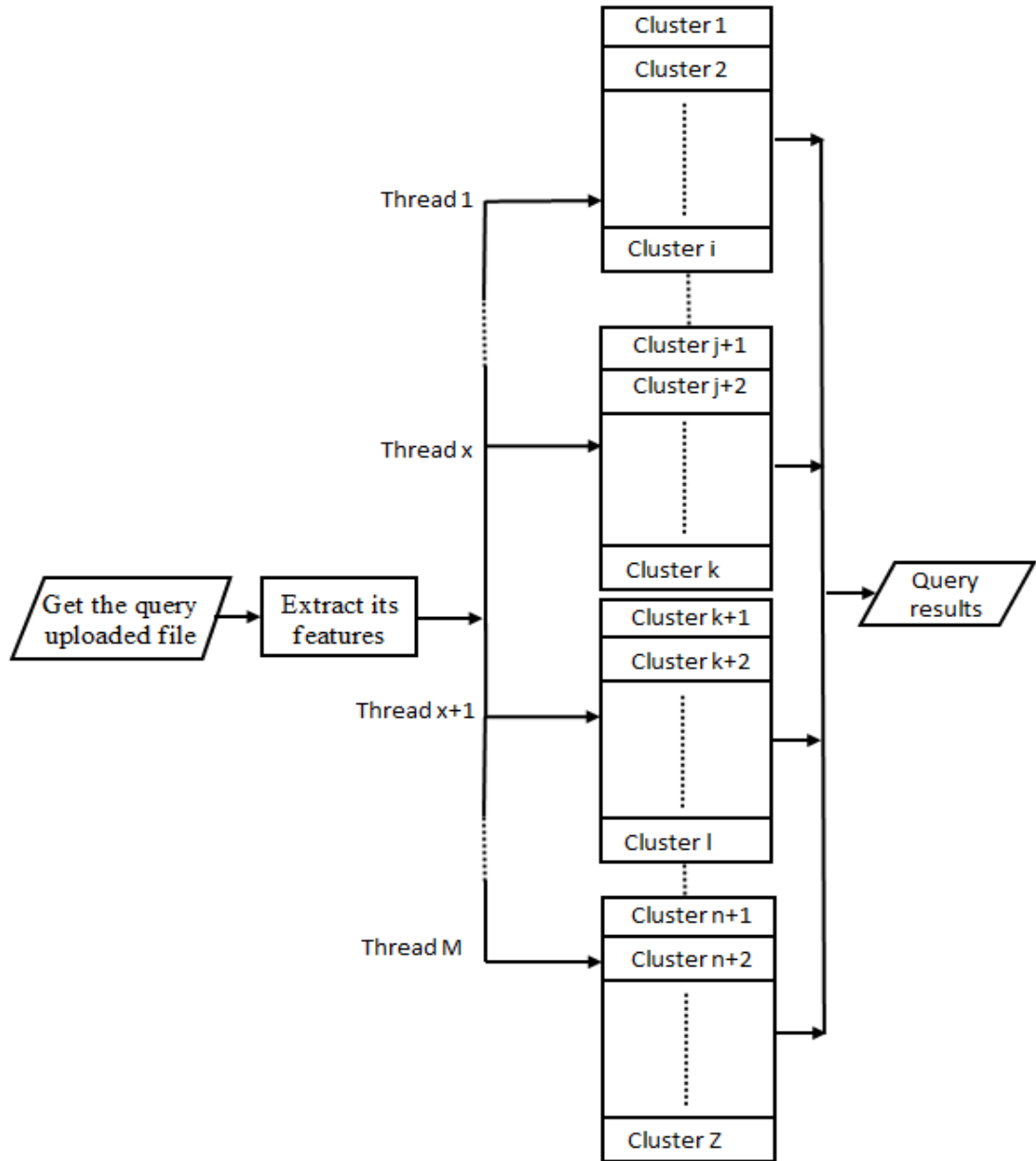


Figure 6.3: The flowchart of parallel implementation of part II of the proposed QBE, using M threads, for Z different clusters.

The total number of comparison for the enhanced QBE, i.e., parallel QBE with clustering implementation, has been compared with the following two scenarios:

- A. Sequential QBE without clustering.
- B. Sequential QBE with clustering.

Assuming that we have N files in the database(s), K clusters, and P threads, then the total number of comparison for these approaches is shown in Table 6.1.

Table 6.1: Total number of comparison of the enhanced QBE as compared with A, B scenarios.

Scenario	The proposed parallel QBE with clustering	A	B
Total number of comparisons	K/P	N	K

It is important to note that in the case of QBE with clustering (the enhanced QBE and scenario B), the comparison is done only with the leader of each existing cluster, while, in the case of sequential QBE (scenario A), the comparison is done with all database(s) files.

6.2.3 Elimination of Duplicated Multimedia Files

In the enhanced QBE, the elimination of the duplicated files is performed during the process of adding any new file to the database. This process is done by adding a flag field to the multimedia database table(s). This flag is set to 1 for a file and will be set to 0 for its duplicated ones (if any). As the enhanced QBE is dealing with clusters, the elimination of the duplicated files is done by applying the following two steps:

- A. Find the most similar cluster(s): This is done by comparing the new file with the leaders of the existing clusters.
- B. Compare the new file with the files in the most similar cluster(s): In this case, if a duplicated copy found in any of these clusters, the new file flag will be set to 0, and to one otherwise.

6.3 Query Processing Mechanism

During the query, the features of the uploaded file will be extracted, and its similarity with the clusters leaders will be found using Euclidean distance formula. The contents of the most similar cluster(s) will be shown to the user. The flowchart of the query process unit in the enhanced QBE can be seen in Figure 6.5.

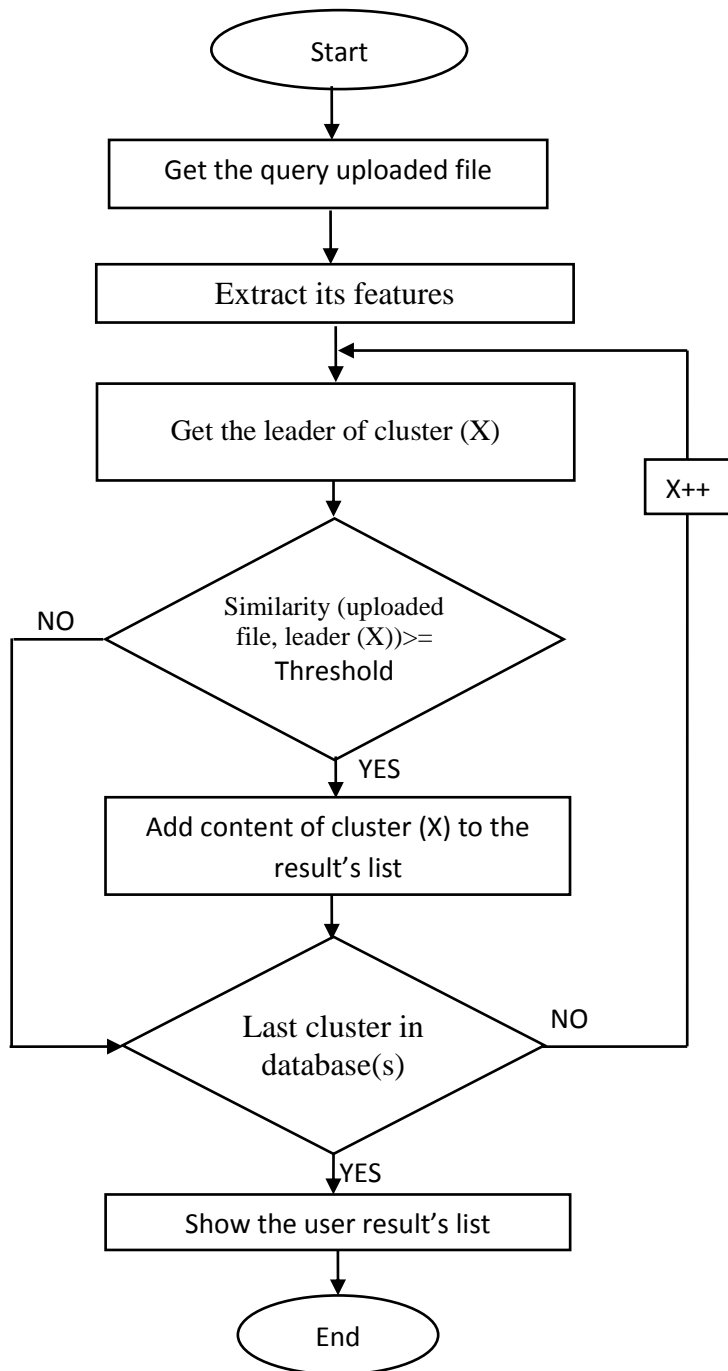


Figure 6.4: The flowchart of the query process unit in the proposed QBE.

6.3.1 User Options for the Queries Using the Proposed QBE

A. Image QBE Options

- 1) Find exactly similar images to the uploaded one.
- 2) Find all images that have a specific percentage of similarity to the uploaded one. For instance, if the user selects 50% as a percentage of similarity, the enhanced QBE will find all images in the database that have at least half of its pixels similar to the uploaded file.

B. Video/Audio QBE Options

As the video and audio files have approximately the same structure, the following options are used to deal with these files:

- 1) Find exactly similar files to the uploaded one.
- 2) Get the complete file for an available part of video or audio.
- 3) Find a higher quality video or audio file that can be of different format, i.e., extension.
- 4) Find all files which have a specific percentage of similarity to the uploaded one.

It is important to note that, in case that the user wants to find exactly similar files to the uploaded one, the enhanced QBE will ignore the flag of database files.

Chapter 7

EXPERIMENTAL STUDY

In this study, several experiments have been conducted to investigate the performance of the proposed WBC, the re-ranking, the elimination of the duplicated multimedia files and the enhanced QBE approaches. The number of days and the number of samples used in these experiments were chosen to achieve a 95% confidence level.

7.1 Crawler Technique Performance

The performance of the conventional crawler, and the proposed WBC for static and dynamic websites was studied in seven experiments. In the WBC experiments the time-based-flag mechanism was used to re-set the flag in the watcher report. The server PC used in this study has the following characteristics: Intel Core i7 CPU, 2.4 GHz clock frequency, 8GB RAM, 1TB Hard Drive, and the operating system was Windows 7.

Experiment 1: Percentage of updated webpages in a group of websites

In this experiment, the watcher file has been used to monitor a group of websites that belong to different categories, such as education, sport, banks and news, for one week and the number of updated pages in each website was recorded and shown in Table 7.1. The average percentage of the updated and the newly added pages was calculated using Eq.7.1.

$$\text{Average \%} = \frac{1}{7} \sum_{D=1}^7 \frac{N(D)}{M} \times 100\% \quad \text{Eq.(7.1)}$$

where N represents the number of updated and newly added webpages in a specific day, M represents total number of webpages. This experiment was performed by downloading a copy of the monitored websites pages. Then, a copy of the watcher file was run for each website to detect the updated and the newly added pages. The following can be seen from Table 7.1: First, less than 12% of the webpages are updated. This means that the conventional crawling techniques are spending most of their working time in visiting the un-updated pages. Second, the percentage of the updated webpages in websites 6 and 7 that belong to news category was very small. This is due to the fact that such websites have to keep the old information available to the users, and the update process is done by adding new webpages.

Table 7.1: The number of updated pages for different websites recorded in seven days.

Website #	Website URL	Total number of webpages (M)	Number of updated and newly added webpages (N)							Average %
			Day1	Day2	Day3	Day4	Day5	Day6	Day7	
1	cmpe.emu.edu.tr	≈591	74	70	60	77	65	63	62	≈11.5%
2	ahu.edu.jo/colleges/engineering	≈402	43	44	40	50	35	39	34	≈10%
3	global.nytimes.com	≈15090	1346	1403	1321	1297	1273	1432	1357	≈9%
4	garanti.com.tr/en	≈153	10	8	11	10	7	9	8	≈6%
5	picoftengineering.com	≈35	2	1	2	2	4	3	2	≈5%
6	Sarayanews.com	≈3377	80	110	89	120	81	94	85	≈3%
7	Koora.com	≈9320	200	150	170	220	190	175	195	≈2%

Experiment 2: Crawler overlapping and communication problems study

In this experiment, the crawler overlapping and communication problems were studied for the developed WBC and for the following three popular crawling scenarios, which we have re-implemented: First, independent crawlers, which means that the working crawlers have no communication between each other and more than one crawler may visit the same website. Second, dependant crawlers, which means that the crawlers have to communicate with each other before visiting a new webpage. Third, seed-server crawler: this scenario eliminates the communication between the running crawlers by introducing a seed-server which is responsible for crawler's decisions such as allowing a crawler to visit a specific website [28].

The experiment was conducted by running fifty parallel copies for each scenario for three days. These crawlers were allowed to crawl all websites under the domain of “*emu.edu.tr*”, and the extracted outer links. Table 7.2 shows the number of times that randomly selected webpages were downloaded for the four scenarios, and Figure 7.1 shows the number of downloaded webpages. From Table 7.2 and Figure 7.1 the followings can be derived:

- 1) In the independent scenario, more than 40 out of 50 crawlers have visited the same webpages. This means that the percentage of the overlapping problem was above 80%, and this decreases the performance of the crawlers. On the other hand, these crawlers were able to download on the average 688,120 webpages.
- 2) Although the dependent scenario has reduced the overlapping problem (less than 7 out of 50 crawlers have visited the same webpages), the average of total number of distinct downloaded webpages was less than 240 thousands.

- 3) The seed-server strategy has successfully solved the overlapping problem. On the other hand, this approach was able to download only around 350,816 distinct webpages.
- 4) Finally, by making use of the flag in the developed WBC there is no communication between the running crawlers, and only one crawler can process a specific page. In addition, the WBC was able to download approximately one million distinct webpages, which is almost double the number of the downloaded pages as compared with the independent strategy, which was found to be the second best scenario. Furthermore, the WBC has downloaded the five pages in the first day, as shown in Table 7.2, and then downloads only the updated pages in the second and third day. Finally, it must be mentioned that the WBC was the only approach which is capable of downloading the updated pages only, while the other scenarios may download updated and un-updated pages.

Table 7.2: The frequency of downloading specific webpages in three days by running fifty parallel crawlers.

Web page #	Frequency											
	Independent			Dependent			Seed-Server			WBC		
	Day 1	Day 2	Day 3	Day 1	Day 2	Day 3	Day 1	Day 2	Day 3	Day 1	Day 2	Day 3
1	43	44	47	5	7	4	2	1	1	1	0	0
2	44	41	43	4	3	2	1	2	2	1	1	1
3	42	42	45	7	5	4	2	1	2	1	1	1
4	47	45	43	1	7	3	1	2	1	1	0	0
5	44	43	40	7	5	4	1	2	3	1	0	0

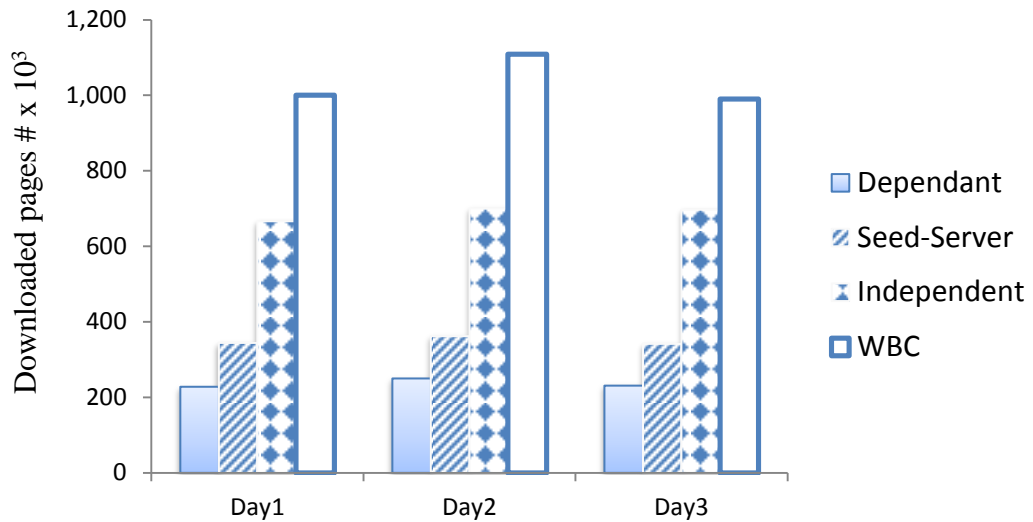


Figure 7.1: The total number of downloaded webpages using dependant, seed-server, independent strategies, and the developed WBC in three days.

Experiment 3: Watcher file effects on the site servers

In this experiment, the effects of the watcher file on the websites server have been investigated by recording the size of the allocated memory and the percentage of CPU usage for the watcher file, when it is uploaded and run on a server that contains 10 thousands, 100 thousands, and 1 million webpages. The results are recorded and shown in Table 7.3.

Table 7.3: The watcher file requirements.

Number of webpages	Size of allocated memory	Percentage of CPU usage
10,000	62.351 KB	0.84%
100,000	155.857 KB	2.37%
1,000,000	197.140 KB	3.45%

It is clear from Table 7.3 that the watcher file does not require large CPU and memory resources. For instance, only 3.45% of CPU usage and around 197 KB of the memory were used for 1 million webpages.

Experiment 4: Number of distinct downloaded webpages study

This study is composed of two experiments. In the first experiment, the performance of the proposed WBC and the WEB-SAILOR crawler [28] was studied. We have re-implemented the same scenario of [28]. As the SAILOR crawler requires a seed-server, three machines were used for this approach while two machines were used for the proposed WBC. Two machines were used as crawler clients for both scenarios: one for *.com* domain, and the other for downloading pages from *.edu*, *.net*, and *.org* domains. Due to the huge number of published *.com* websites compared with other domains, the number of threads for the crawler clients has been specified to be 25 for *.com*, and 10 for *.edu*, *.net*, and *.org* [28, 49]. Finally, both crawlers were allowed to crawl a group of randomly selected URLs, and then visit all extracted URLs. The number of downloaded distinct webpages for both approaches was recorded in Table 7.4. It can be seen from Table 7.4 that the proposed WBC increases the number of visited websites by approximately a factor of 3 as compared with the SAILOR crawler.

Table 7.4: Number of downloaded distinct webpages using the proposed WBC and WEB-SAILOR [28].

Day #	Number of downloaded webpages	
	WEB-SAILOR	WBC
1	126,981	381,007
2	131,045	390,074
3	126,987	383,941

In the second experiment, the performance of the proposed WBC was compared with the most popular open source crawlers: Apache Nutch [119], and Scrapy [120]. This experiment was repeated for seven days, where one copy of Apache Nutch, Scrapy

and the WBC was run on a separate machine, and each crawler can run at most fifty threads. The number of webpages that have been processed by these crawlers were recorded in Table 7.5. It is clear from Table 7.5 that the proposed WBC outperforms both the Apache Nutch, and Scrapy crawlers by a factor of approximately 1.6, and 1.4, respectively.

Table 7.5: Number of downloaded webpages using the proposed WBC, Apache Nutch [119], and Scrapy [120].

Day #	Number of downloaded webpages		
	WBC	Apache Nutch	Scrapy
1	113,854	71,154	82,140
2	114,978	70,447	80,261
3	114,864	71,601	81,126
4	115,912	72,008	82,307
5	114,991	70,397	81,151
6	114,784	71,029	81,353
7	115,005	72,042	82,046

Experiment 5: The performance of the re-visiting policies

In this experiment, the re-visiting performance has been investigated for the WBC and the following three re-visiting policies: the uniform [21-25], the proportional by rank [21-23], and the proportional by top N levels [21, 26]. This experiment was repeated for seven days, where one crawler with five threads was used for each policy. In addition, the crawlers were responsible to visit and process news website: *www.hkjtoday.com*. The performance of the re-visiting policies was studied according to the following factors:

- 1) The time required for re-visiting and downloading a website pages.
- 2) The percentage of the updated pages that the crawler was able to download at the time of visiting.

In the proportional policies, the number of levels (N) and the threshold values were selected based on experimental values that provide the best result. The required time for processing the website and the percentage of downloading the updated and newly added pages are shown in Figures 7.2 and 7.3, respectively.

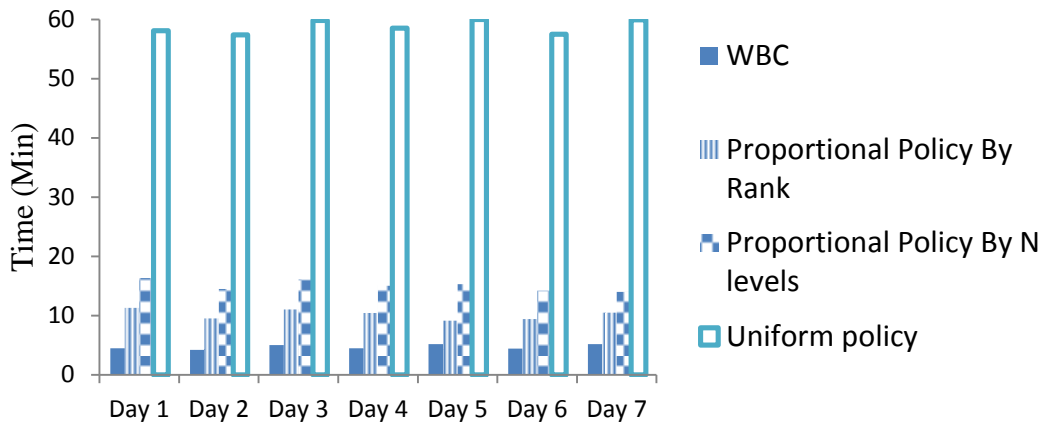


Figure 7.2: The required time for re-visiting “www.hkjtoday.com” website using the crawlers of WBC, uniform, and proportional by rank and by top N level.

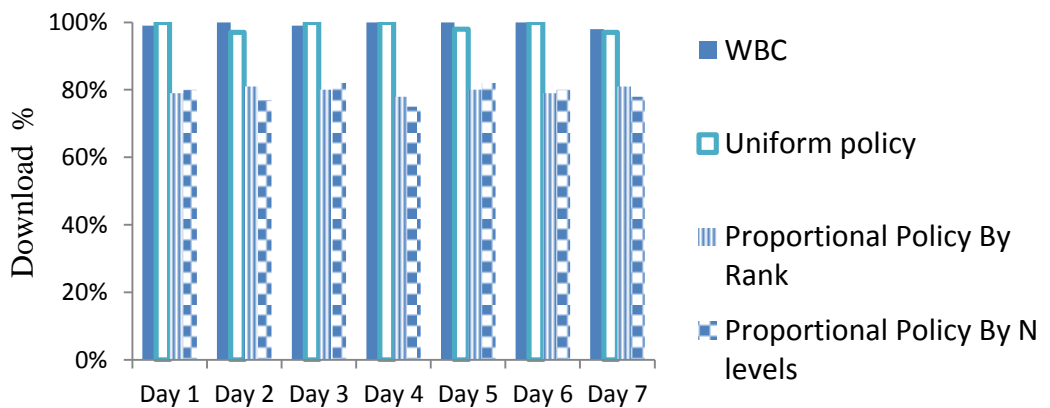


Figure 7.3: The percentage of downloading updated pages in “www.hkjtoday.com” website using the crawlers of WBC, uniform, and proportional by rank and by top N levels.

It can be seen from Figure 7.2 that the proposed WBC decreases the website processing time by a factor of 2.5 as compared with the second best scheme, which is

found to be the proportional by ranking. In addition, the processing time for the uniform policy is approximately 10 times that of the WBC. This is because the uniform policy has to visit and download the entire website. Furthermore, it can be seen from Figure 7.3 that the WBC and the uniform policy were able to download all the updated pages in most cases, whereas the proportional policies were able to download at most 82% of the updated pages. Based on this study, it is clear that the WBC outperform the performance of the other re-visiting policies.

Experiment 6: WBC performance for processing AJAX websites

In this experiment, the performance of the proposed WBC and the crawler developed in [7] for processing AJAX websites were studied. We have repeated the same scenario performed in [7] by using the WBC with 1, 2, 3, and 4 crawlers, where each of the working crawler can run up to four threads. The number of processed dynamic pages in 10 minutes was recorded and shown in Figure 7.4. It can be seen from Figure 7.4 that the proposed WBC increases the number of the processed dynamic pages by approximately 20% as compared with the approach of [7].

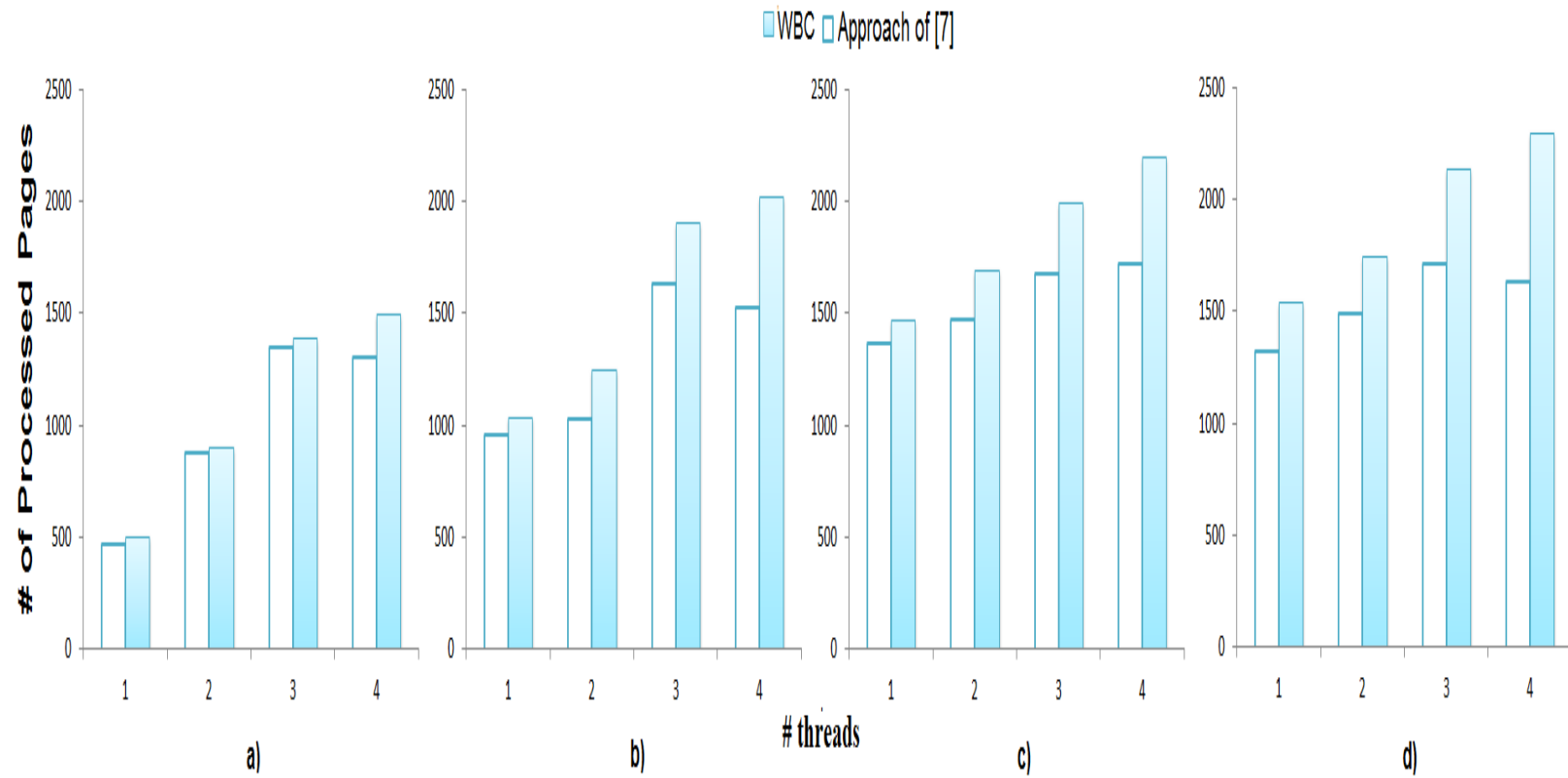


Figure 7.4: Number of dynamic pages processed in 10 minutes for (a) one crawler, (b) two crawlers, (c) three crawlers, and (d) four crawlers.

Experiment 7: WBC performance for real AJAX data set

In this experiment, the performance of the proposed WBC for processing a real AJAX data set was studied. This experiment was performed on YouTube videos data set [121]. The main aim of this experiment is to investigate the ability of the WBC to access and download the related comment pages written by the video viewer. The number of the video comments reported by [121, 122] and the number of comments that the WBC was able to obtain by triggering the related AJAX events are recorded in Table 7.6. It can be seen from Table 7.6 that the proposed WBC was able to download more than 98% of the reported AJAX comment pages.

Table 7.6: Number of comments for specific YouTube videos reported in [121] and downloaded using the proposed WBC.

Video Name	Number of comments		Percentage of the downloaded comments
	Reported in [121]	Downloaded by the WBC	
New Numa - The Return of Gary Broolsma!	17657	17589	99.61%
CRAZED NUMA FAN !!!!	3010	3010	100%
Influence	3866	3862	99.90%
My united states of... WHATEVA !!!	4447	4440	99.84%
Heather Martin - When Are You Coming Home	6045	6044	99.98%
The COMMENT Video!!	976	960	98.36%
iBlinds - New generation iPod accessory	3606	3606	100%
Smosh Short 1: Dolls	7919	7875	99.44%

7.2 Re-ranking Technique Performance

To study the performance of the proposed re-ranking approach, the following six experiments were conducted. The server machine used in these experiments was a PC, with Windows 7 operating system, and with the following properties: Intel Core i3 CPU, 3.3 GHz clock frequency, 4GB RAM, and 512GB Hard Drive.

Experiment 1: Query re-ranking performance using the SCD, EHD, JCD, and the proposed approach

This experiment was conducted by re-ordering the first fifty files of Google results for specific queries using the SCD, EHD, JCD descriptors and the proposed approach. The performance was measured by calculating the *R-Precision* (R_{Prec}) [123], which is the precision value obtained for the top retrieved R documents, and computed as

$$R_{Prec} = \frac{r}{R} \quad \text{Eq.(7.2)}$$

where r is the number of relevant documents among the top R retrieved ones. In this experiment, the precision was found for $R=10$. Table 7.7 shows the 10-Precision of the SCD, EHD, JCD, and the proposed approach for re-ranking the results of specific queries. In the case of the proposed approach, the effect of assigning equal or dynamical weights for the used descriptors has also been investigated. It can be seen from Table 7.7 that the proposed approach was able to show more relevant files to the top of the list than the SCD, EHD, and JCD descriptors. In addition, it is clear that assigning the descriptors weights dynamically outperform the equal weights assignment.

Table 7.7: The 10-Precision of the SCD, EHD, JCD, and the proposed approach for re-ranking image and video queries.

Query type	Query	SCD	EHD	JCD	The Proposed Approach	
					Equal Weight	Dynamic Weight
Images	World Cup	0.2	0.4	0.5	0.6	0.7
	Jordan Country	0.7	0.8	0.8	0.8	0.9
	white pages	0.3	0.7	0.7	0.8	0.9
	EMU Logo	0.5	0.6	0.6	0.6	0.7
	Ebola	0.4	0.3	0.4	0.5	0.6
	Malaysia Airlines	0.4	0.4	0.5	0.6	0.7
	Frozen	0.6	0.5	0.6	0.7	0.8
	Sochi Olympics	0.5	0.5	0.5	0.6	0.6
Videos	Mutant Giant Spider Dog	0.4	0.3	0.5	0.7	0.9
	World Cup	0.3	0.3	0.3	0.4	0.5
	EMU	0.6	0.6	0.7	0.8	0.8
	Ebola	0.2	0.4	0.5	0.5	0.6
	Jordan	0.6	0.5	0.5	0.7	0.8
	Malaysia Airlines	0.7	0.7	0.7	0.8	0.9
	Frozen	0.3	0.5	0.5	0.6	0.7
	Sochi Olympics	0.2	0.3	0.3	0.5	0.7

Experiment 2: Position of the most relevant files for image and video queries

This experiment includes two parts. In the first part of this experiment, the position of the first five relevant files for specific queries using Google, Yahoo search engines was found. The results of this experiment are shown in Table 7.8. Two independent evaluators were used to decide on the file relevancy. It can be seen from Table 7.8

that the relevant files are distributed among large number of results. This leads the user to check a large number of results to get the required files and it is time consuming process.

In the second part of the experiment, the position of first ten relevant files for the same queries is obtained after re-ordering the first hundred files of the Google results. These results were computed using JCD, which is found to be the best among the other descriptors, and the proposed re-ranking approach (PRA). The results are shown in Table 7.9. It is clear from Table 7.9 that the proposed re-ranking approach outperforms the JCD scheme. In addition, the proposed re-ranking approach successfully re-orders the original results and shows the most relevant ones at the top of the list.

Table 7.8: The position of first five relevant files for specific queries obtained using Google, and Yahoo search engines.

Query type	Query	Google					Yahoo				
		1 st relevant	2 nd relevant	3 rd relevant	4 th relevant	5 th relevant	1 st relevant	2 nd relevant	3 rd relevant	4 th relevant	5 th relevant
Images	World Cup	10	22	25	33	50	10	29	33	42	63
	Jordan	3	4	10	32	55	4	8	14	27	41
	pages	5	20	30	55	65	25	30	35	50	61
	EMU Logo	1	3	15	37	82	8	9	25	67	81
	Ebola	5	27	28	31	54	33	37	45	51	65
	Malaysia Airlines	31	34	53	79	131	6	9	39	52	63
	Frozen	3	5	20	34	35	12	20	27	30	34
	Sochi Olympics	13	22	24	42	52	4	13	17	34	41
Videos	Mutant Giant Spider Dog	1	5	9	17	20	2	9	16	18	25
	Baby	1	2	27	32	57	1	14	20	26	32
	World Cup	3	6	14	22	25	9	17	24	37	47
	Jordan	2	6	16	24	31	10	32	40	68	68
	Ebola	6	14	29	43	61	3	14	27	45	59
	Malaysia Airlines	3	16	31	38	42	15	33	45	57	58
	Frozen	6	22	27	34	41	3	18	20	24	30
	Sochi Olympics	1	24	25	32	40	2	11	20	29	32

Table 7.9: The position of first ten relevant files for specific queries using JCD, and the proposed re-ranking approach (PRA).

Query type	Query	1 st relevant		2 nd relevant		3 rd relevant		4 th relevant		5 th relevant		6 th relevant		7 th relevant		8 th relevant		9 th relevant		10 th relevant	
		JCD	PRA	JCD	PRA	JCD	PRA	JCD	PRA	JCD	PRA	JCD	PRA	JCD	PRA	JCD	PRA	JCD	PRA	JCD	PRA
Images	World Cup	2	1	3	2	6	3	7	4	10	5	12	6	13	7	17	8	20	9	21	10
	Jordan	3	1	5	2	7	3	10	4	14	5	15	6	16	8	19	9	22	10	25	11
	white pages	1	1	2	2	5	4	8	6	12	8	14	10	15	11	18	12	21	13	23	14
	EMU Logo	3	1	5	2	8	3	11	4	15	8	18	13	21	15	22	17	26	18	27	19
	Ebola	2	1	4	2	6	3	9	4	13	5	15	6	19	8	21	10	22	11	23	12
	Malaysia Airlines	4	1	5	2	7	4	10	5	15	6	19	7	20	8	24	10	26	11	28	12
	Frozen	2	1	3	2	6	3	9	4	10	5	13	6	15	7	19	8	24	9	25	10
	Sochi Olympics	6	1	7	2	9	3	13	4	17	6	18	7	22	11	23	15	25	16	28	18
Videos	Mutant Giant Spider Dog	4	1	6	2	7	3	8	4	13	8	16	10	20	14	24	16	30	17	35	22
	Baby	6	1	9	2	11	3	13	4	15	5	17	7	22	11	23	12	31	14	38	16
	World Cup	3	1	5	2	7	3	9	4	13	9	15	10	18	11	21	12	28	13	33	14
	Jordan	4	1	5	2	6	3	8	7	12	8	17	11	21	13	24	15	32	16	37	20
	Ebola	6	3	9	4	11	6	13	8	16	10	19	12	24	13	30	16	35	18	41	19
	Malaysia Airlines	5	1	6	2	7	3	9	4	11	5	15	8	19	9	24	11	29	13	35	14
	Frozen	4	1	7	3	10	4	11	7	14	9	17	13	20	14	25	15	29	17	33	18
	Sochi Olympics	3	1	5	3	6	4	9	5	11	6	15	8	19	13	23	15	30	17	37	18

Experiment 3: The effects of multimedia pre-processing on the proposed approach

In this experiment, the effects of the pre-processing on the performance of the developed approach were investigated. This experiment was conducted by downloading the first fifty files of Google results for some video and image queries, and the first thirty files of “Amazon MP3” results for some audio queries. The offline operations were done for the following two scenarios:

- 1) Extracting and saving the features of the files without pre-processing.
- 2) Extracting and saving the features after pre-processing these files.

The *R-Precision* was calculated for $R=10$ by selecting a target file for each query, and then re-ordered the query files. Table 7.10 shows the 10-Precision of the proposed approach with and without the pre-processing operation for the image, video and audio queries. It can be seen from Table 7.10 that the pre-processing operation improves the proposed approach and significantly increases the percentage of the retrieved relevant files. In addition, as mentioned in Chapter 4, Section 4.3.1.1, in some cases the features of two similar audio files may show that these files are different, and this makes the pre-processing an essential operation for the audio files. Furthermore, the pre-processing can decrease the overall time required for indexing and retrieving the multimedia files, especially for video files. For example, for indexing and retrieving a video file with size of 500 MB and with a duration of one hour, it has been found that the proposed pre-processing operations allow 50% saving in the processing time.

Table 7.10: The 10-Precision of the proposed approach with and without the pre-processing operation for re-ranking image, video and audio queries.

Query type	Query#	The Proposed Approach	
		Without pre-processing	With pre-processing
Images	1	0.8	1
	2	0.6	0.9
	3	0.7	0.9
	4	0.6	0.9
	5	0.7	0.8
Videos	1	0.7	0.9
	2	0.6	0.9
	3	0.6	0.8
	4	0.5	0.7
	5	0.6	0.9
Audios	1	0.5	0.8
	2	0.6	0.9
	3	0.4	0.8
	4	0.7	0.9
	5	0.5	0.9

Experiment 4: Audio signature performance

In this experiment, the performance of the proposed approach for re-ranking the results obtained from “Amazon MP3” search engine was studied for two cases: single and three dynamic weights signatures. The performance was observed by calculating the 10-Precision by downloading and re-ordering the first thirty files of “Amazon MP3” results. The results are shown in Table 7.11 and it can be seen that the proposed approach was capable of re-ordering the “Amazon MP3” results and show more relevant files at the top of the queries results. In addition, it can also be seen from Table 7.11 that representing each file by three dynamic weighted signatures outperform the performance of using a single signature.

Table 7.11: 10-Precision of “Amazon MP3”, and the proposed approach using a single, and three dynamic signatures.

Query Category	Query Keywords	Amazon MP3	The Proposed Approach	
			Single signature	Three dynamic Signature
Languages	Learn English	0.7	0.7	0.9
	English Numbers	0.4	0.5	0.7
	English alphabet	0.8	0.6	0.9
Baby	Baby Laughing	0.6	0.5	0.7
	Baby Crying	0.4	0.6	0.8
Animals	Funny animal	0.7	0.7	0.8
	lion roar	0.3	0.5	0.7
	bird voices	0.2	0.4	0.6

Experiment 5: Performance of the developed re-ranking approach for a real large-scale multimedia dataset

In this experiment, the performance of the developed re-ranking approach for a real large-scale multimedia dataset was investigated. The multimedia dataset includes 25,000 images, 10,000 audios, and 10,000 videos. In this experiment: 25 image queries, 25 video queries, and 25 audio queries have been conducted. The *R-Precision* for the developed re-ranking approach was calculated by selecting a target file for each query. The precision was computed for R=10, R=20, and R=30. Figures 7.5, 7.6 and 7.7 show the R-precision for image, video and audio queries. Tables 7.12, 7.13 and 7.14 show samples of these queries. It is clear from this experiment that even when the database contains a large number of files, which includes different subjects, the developed re-ranking approach successfully showed the most

relevant files to the top of the query results and therefore, increases the percentage of the retrieved relevant files.

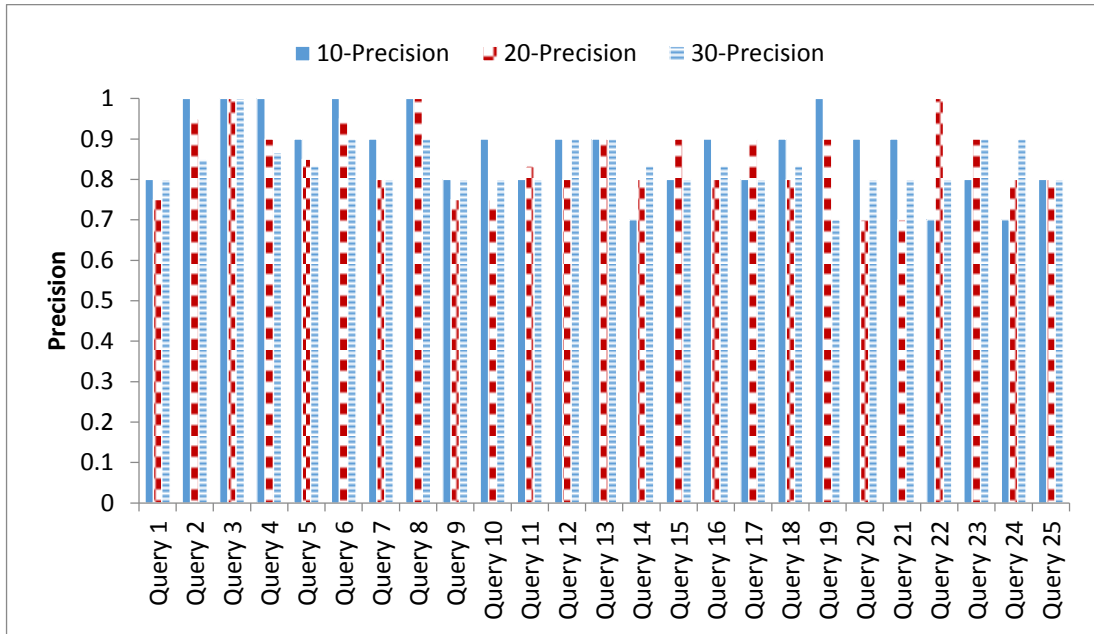


Figure 7.5: The R-Precision for the image queries.

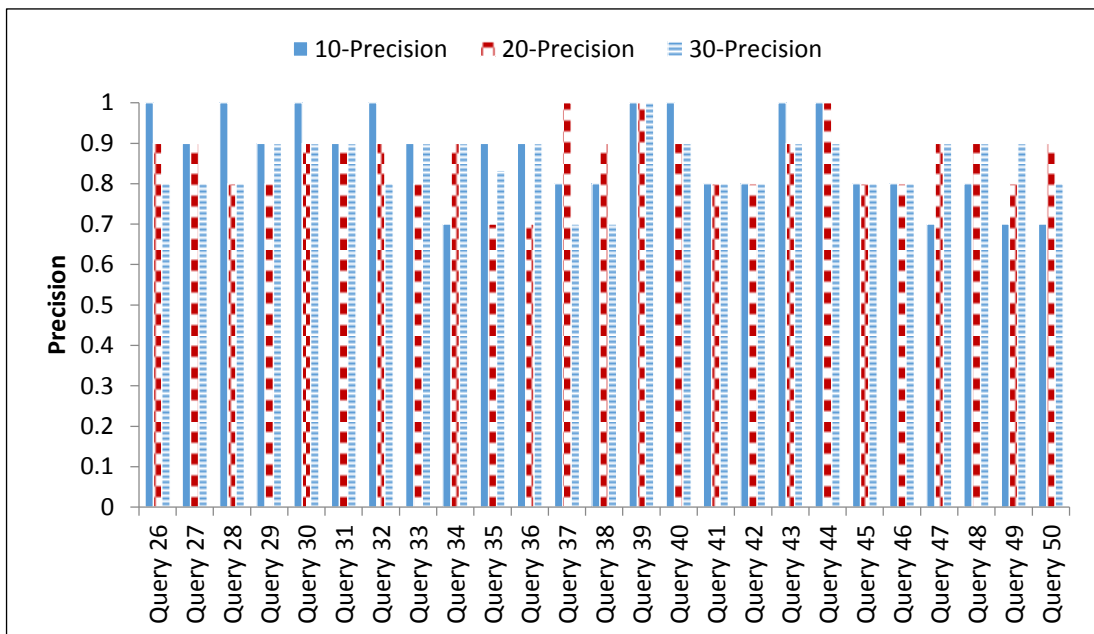


Figure 7.6: The R-Precision for the video queries.

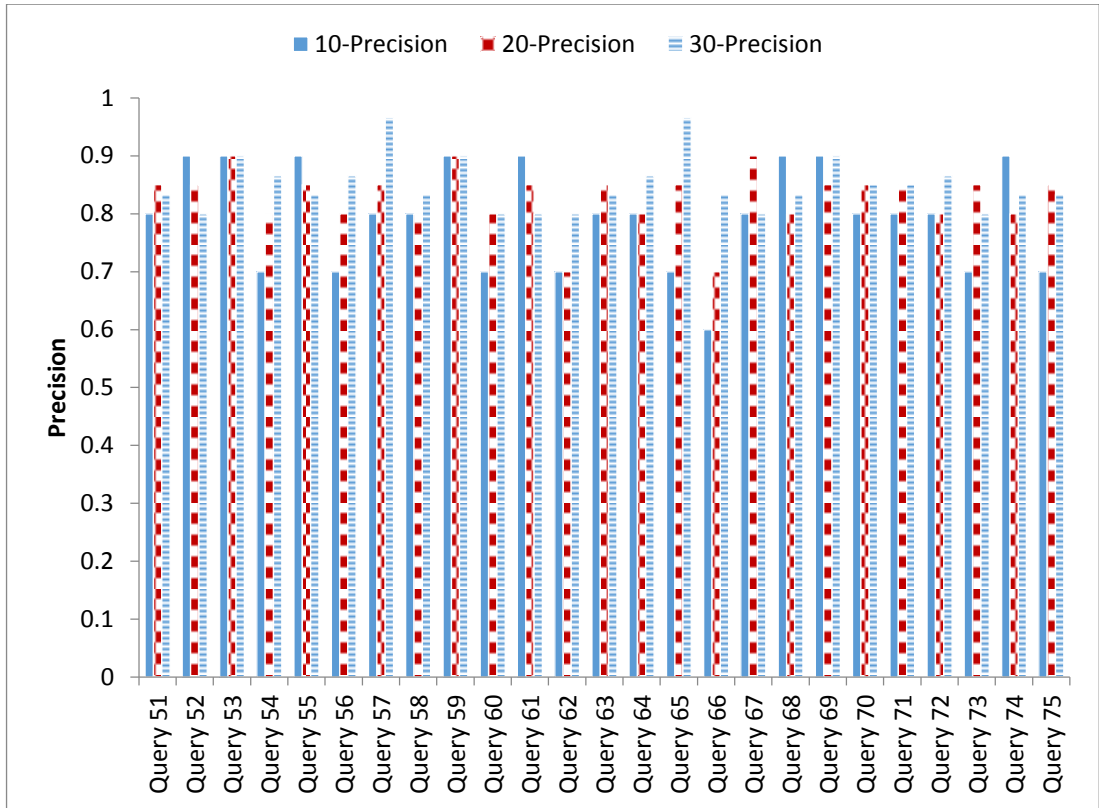


Figure 7.7: The R-Precision for the audio queries.

Table 7.12: R-Precision of the developed re-ranking approach for a sample of the image queries.







Query #	Query Keywords	Query Target	Precision		
			R=10	R=20	R=30
1	Four Seasons of Birds		0.8	0.75	0.8
2	Doors		1	1	1
5	Handicraft Family		0.9	0.85	0.833
10	World Cup		0.9	0.75	0.8
15	Malaysia Airlines		0.9	0.9	0.8
25	Sochi Olympics		0.8	0.8	0.8

Table 7.13: R-Precision of the developed re-ranking approach for a sample of re-ranking video queries.





Query #	Query Keywords	Samples of the Target(Video) Key Frame	Precision		
			R=10	R=20	R=30
27	EMU		0.9	0.8	0.9
31	Frozen		0.9	0.9	0.9
38	Sochi Olympics		0.8	0.9	0.8
49	Mutant Giant Spider Dog		0.7	0.8	0.9

Table 7.14: R-Precision of the developed re-ranking approach for a sample of re-ranking audio queries.

Query #	Query Keywords	Precision		
		R=10	R=20	R=30
51	Learn Spanish	0.8	0.85	0.833
59	Baby Laughing	0.9	0.9	0.9
64	Lion roar	0.8	0.8	0.833
68	Learn English	0.9	0.85	0.8
75	Gangnam Style	0.9	0.85	0.833

Experiment 6: Performance of the developed re-ranking approach versus the approaches of [34] and [41]

In this experiment the performance of the developed re-ranking approach was compared with that of [34] for image re-ranking and with that of [41] for video re-ranking. The dataset of experiment 5 have been used for all studied approaches. Note that we have re-implemented the main algorithm of [34] and [41], and we have used their tuned parameters. The *R-Precision* was found for R=10, R=20, and R=30. Figures 7.8 and 7.9 show, respectively, the performance of the proposed re-ranking approach as compared with that of [34] for images, and with that of [41] for videos. From these Figures, it is obvious that the proposed re-ranking approach outperform both the approaches of [34] and [41] for images and videos, respectively.

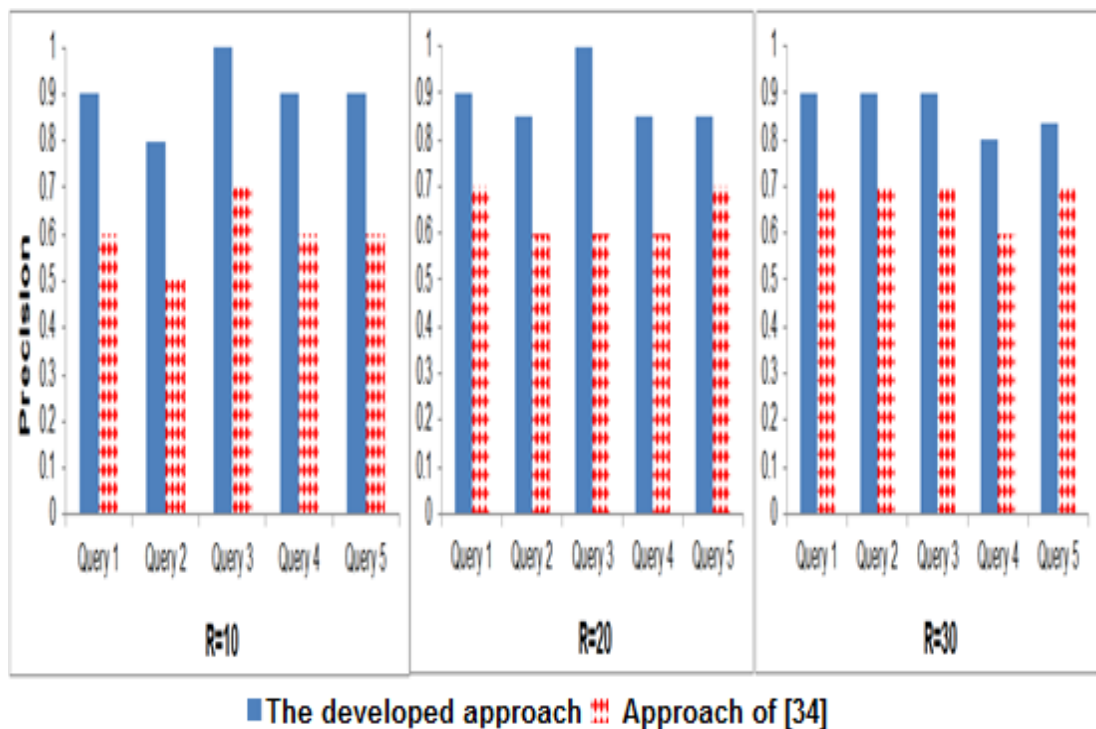
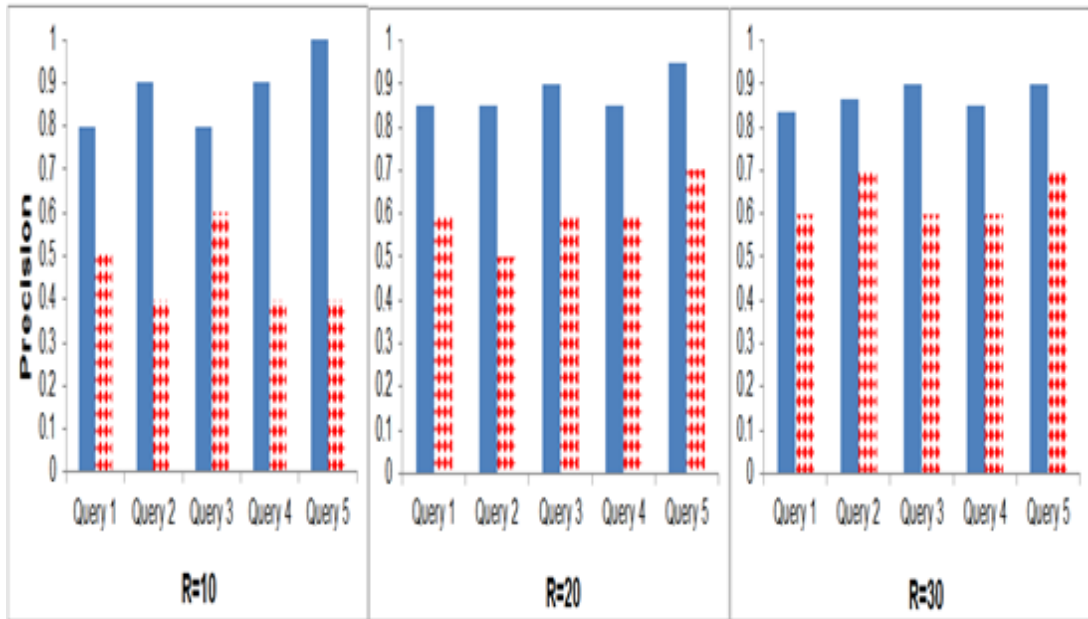


Figure 7.8: R-Precision of the developed re-ranking approach and the approach presented in [34] for specific image queries.



■ The developed Approach ❦ Approach of [41]

Figure 7.9: R-Precision of the developed re-ranking approach and the approach presented in [41] for specific video queries.

7.3 Elimination Technique Performance

In this section, the performance of the proposed elimination scheme described in chapter 5 was investigated in seven experiments. These experiments were conducted using image, video, and audio databases. These databases have been created by randomly taking multimedia files from the Internet. The server used in these experiments has the following characteristics: Intel Core i7 CPU, 4.0 GHz clock frequency, 16GB RAM, 256GB SSD, 1TB Hard Drive, and the operating system was Windows 7.

7.3.1 Image Database Processing

In this type of database, the Bit-wise [13] and the MD5 hashing [16, 97] algorithms have been used for the comparison process. In the Bit-wise scheme, the comparison is done for all pixels in the images. On the other hand, in the MD5 scheme, the comparison is limited to 16 Bayt. The performance of the proposed elimination scheme for image database processing was studied in two experiments.

Experiment 1: Performance of the Bit-wise versus the MD5 algorithms

In this experiment, the execution time versus number of images for the two algorithms has been measured and shown in Figure 7.10. From this Figure, it is clear that the Bit-wise algorithm is not efficient for large image databases. On the hand, the MD5 algorithm was found to be much faster than the Bit-wise, especially for large image databases. In the following experiments the performance of parallel implementation of MD5 will be investigated.

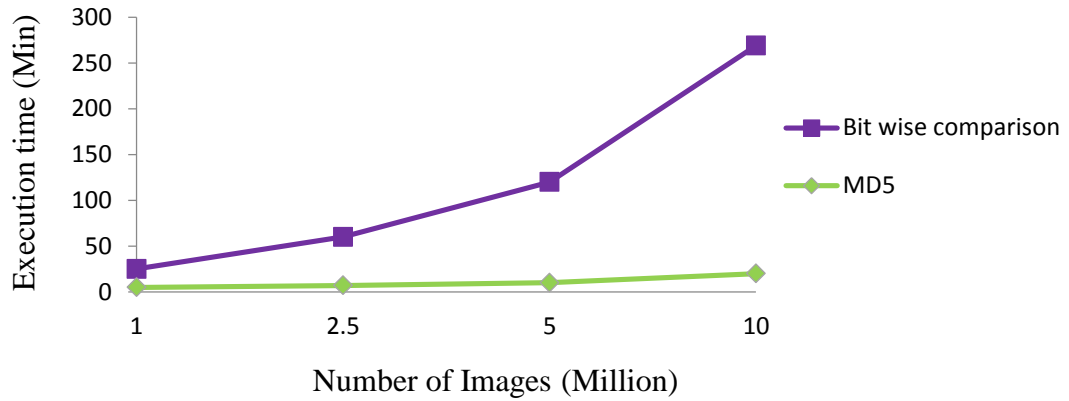


Figure 7.10: Execution time versus number of images for the MD5 and Bit-wise algorithms.

Experiment 2: Performance of parallel implementation of the MD5 algorithm

In this experiment, the performance of the parallel implementation of MD5 algorithm is investigated by observing the execution time versus the number of images when running 4, 8, 12, and 16 processes. The results of this experiment are shown in Figure 7.11. As expected, it can be seen from Figure 7.11 that by using the parallel technique the search time decreases significantly.

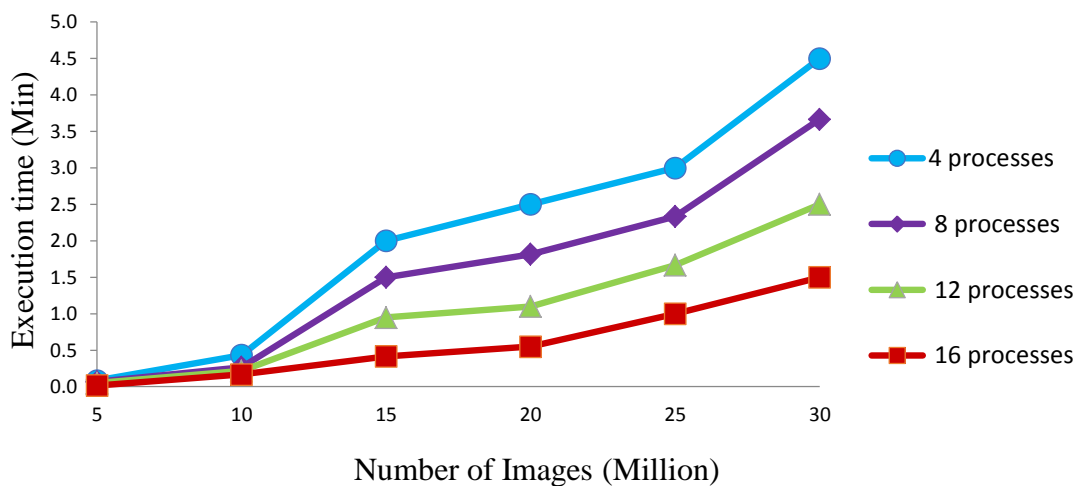


Figure 7.11: Execution time versus number of images for 4, 8, 12, and 16 processes as obtained by using the parallel implementation of the MD5 algorithm.

7.3.2 Video and Audio Databases Processing

For video and audio databases, the MD5 hashing [16, 97] and the low level feature extraction [17] algorithms have been used in the comparison process. The performance of the proposed elimination scheme processing was studied in four experiments.

Experiment 1: Performance of the parallel implementation of the MD5 hashing algorithm for video and audio databases

This experiment has been performed for 4, 8, 12, and 16 processes. The execution time for processing different number of video and audio files is shown in Figure 7.12. It is clear from Figure 7.12 that the MD5 algorithm is capable of dealing with video and audio within acceptable processing times. For instance, by running 16 processes, a database that contains 30 million multimedia files can be processed in less than two minutes. On the other hand, we have observed that MD5 hashing algorithm was not able to detect duplicated video and audio files that have different formats, i.e., extensions. Therefore, the low-level feature extraction [17] was adapted in the comparison process, and its performance was examined in the following experiment.

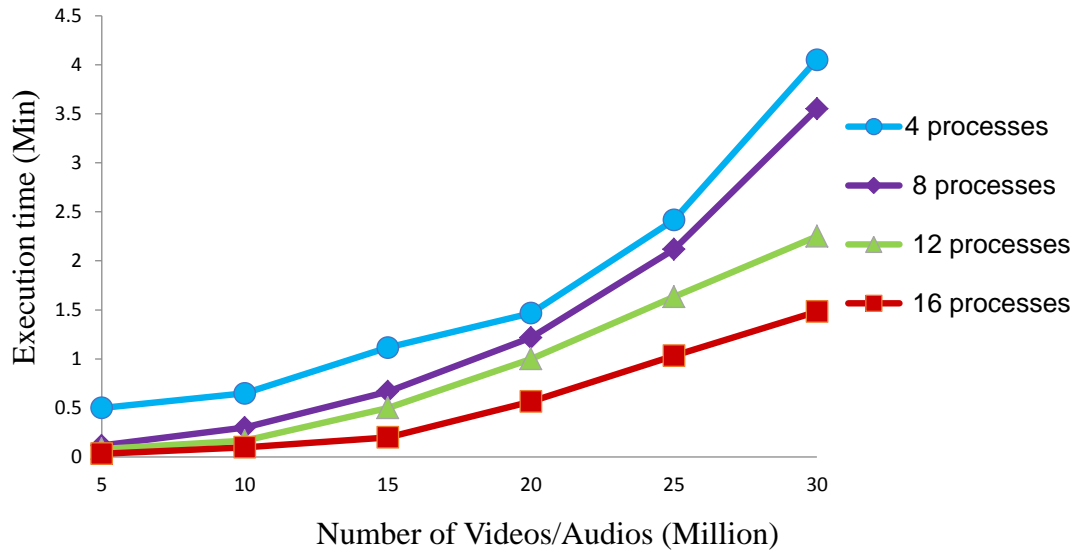


Figure 7.12: Execution time versus number of video and audio files for 4, 8, 12, and 16 processes as obtained by using the parallel implementation of the MD5.

Experiment 2: Performance of the low-level feature extraction versus the MD5 hashing algorithm

In this experiment, the performance of low-level feature extraction and MD5 algorithms has been studied according to the following two factors:

- 1) The time required for creating video/audio databases. In this step, it is important to note that the files features and the hash values were extracted and saved in the databases without performing the comparison process, i.e., the flag of the files is not set in this step.
- 2) The time required for the comparison process, i.e., detecting the duplicated video/audio files and setting the file's flag.

The results of this experiment are shown in Figures 7.13, 7.14, and 7.15. It can be seen from Figure 7.13 that the time required for creating video/audio databases using MD5 hashing algorithm is larger than the time required using the low level extraction technique. On the other hand, the MD5 algorithm was found to be faster in the

comparison process as shown in Figure 7.14. Figure 7.15 shows the total execution time required for the MD5 algorithm and low level extraction. From this Figure, it is clear that both techniques have approximately the same performance.

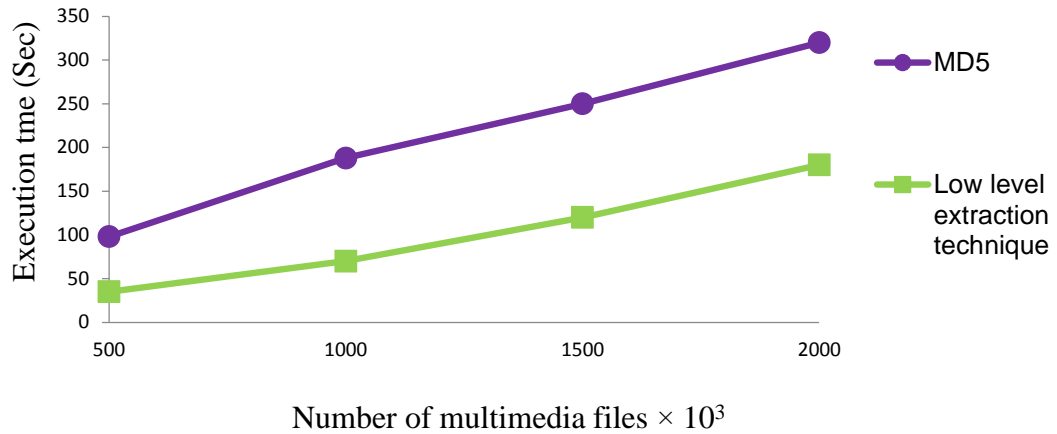


Figure 7.13: Execution time required for creating video/audio databases using the low level extraction and the MD5 hashing algorithm.

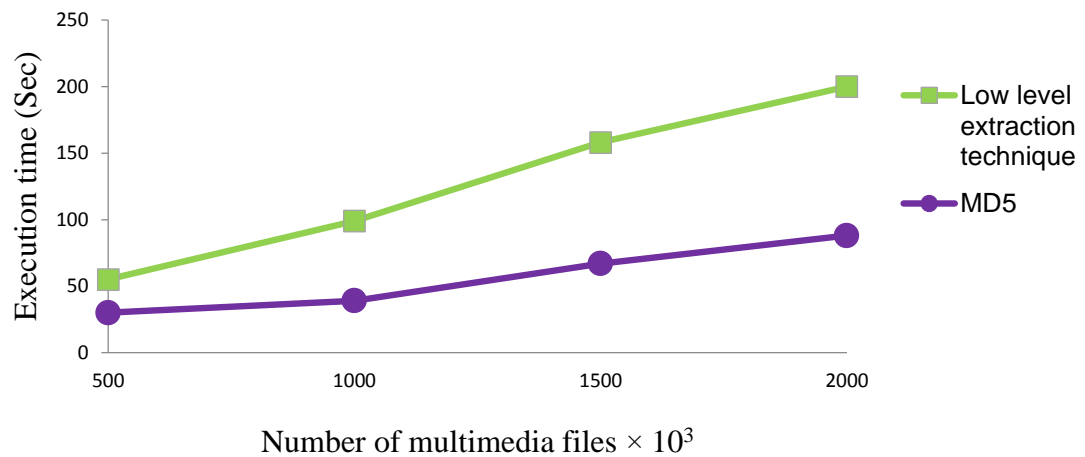


Figure 7.14: Execution time required for the comparison process by using the low level extraction and the MD5 hashing algorithm.

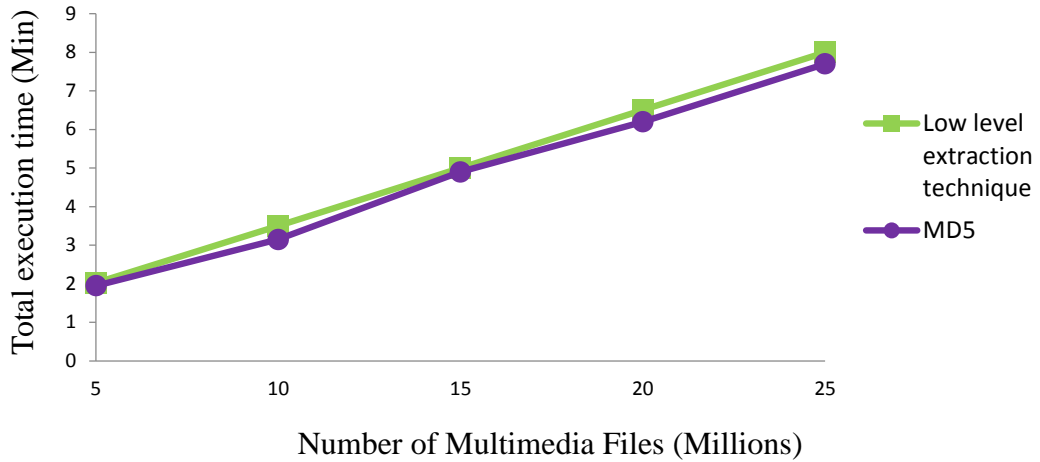


Figure 7.15: Total execution time versus number of file using the low level extraction and the MD5 hashing algorithm.

Experiment 3: The required time for updating multimedia databases

In general, the contents of most websites are updated and modified in a routine manner. Consequently, the search engines have to update their databases in a similar manner. We have performed this experiment to investigate the time required for the comparison during the process of updating a database (adding new files to an existing database). Table 7.15 shows the execution time for updating three databases that contain 10, 100 and 1000 million files. It is clear from Table 7.15 that the proposed elimination scheme, not only has the ability of updating large databases, but also eliminates duplicated multimedia files.

Table 7.15: Execution time for updating a database using the proposed elimination scheme.

Number of files in the database (Million)	Number of new files to be added to the database (Million)	Time (Min)			
		32 Processes	64 Processes	128 Processes	256 Processes
10	5	1	0.45	0.23	0.15
	7,5	1.89	0.98	0.51	0.27
	10	2.2	1.23	0.69	0.41
	15	3.2	1.7	0.91	0.57
	50	8.91	4.76	2.67	1.41
100	5	4	2	1	0.5
	7,5	6	3	1.6	0.8
	10	9	4.8	2.5	1.8
	15	16	8.7	4.7	2.5
	50	81	42.5	23	11
1000	5	35.3	18	10	4.9
	7,5	52.7	27.1	14.6	8
	10	68.2	37.9	18.4	10.1
	15	101	50.5	26.1	14
	50	220.7	113	59	30.8

Experiment 4: The efficiency of Google, Yahoo and Bing search engines with and without using the proposed elimination scheme

In this experiment, the efficiency of Google, Yahoo and Bing search engines with and without using the proposed elimination scheme was studied. In this experiment, the queries were performed for the most frequently searched keywords, which we have obtained from [124, 125]. The percentage of relevant files and the percentage of duplicated files for these queries using Google, Yahoo and Bing search engines have been recorded in Tables 7.16, 7.17 and 7.18. It is important to note the following: First, the number of results for each query was provided by Google, Yahoo and Bing search engines. Second, the percentage of duplicated files in the queries results was

found by the proposed elimination scheme. Finally, the percentage of the relevant files was found by two evaluators through checking the first 100 files of each query results. It can be seen from these Tables that the queries results obtained by Google, Yahoo and Bing Search engines contain around 40% duplicated files. On the other hand, by integrating the proposed elimination scheme all duplicated files were eliminated and this increases the chance of getting more relevant file.

Table 7.16: The percentage of relevant and duplicated files for Google search engine with and without using the proposed elimination scheme.

Keywords	Google Without using the proposed elimination scheme		Percentage of relevance in the first 100 files of Google query results	
	Number of results (Millions)	Percentage of duplicated files	Without using the proposed elimination scheme	With the proposed elimination scheme
Images				
Hotels	140	35%	88%	91%
Cars	158	32%	92%	95%
Dog	130	39%	89%	94%
Girls	209	40%	91%	97%
Weather	93,6	39%	88%	92%
Videos				
Videos	78,6	30%	92%	96%
Funny videos	16,2	33%	80%	85%
Dance movies	3,7	32%	82%	87%
Movie trailers	3,7	29%	80%	84%
Weight loss	0.42	25%	80%	88%
Audios				
Music	14,8	38%	85%	89%
Songs	6,31	41%	88%	92%
Albums	4,51	40%	80%	86%
MP3	5,52	39%	84%	89%
Clips	5,6	35%	82%	86%

Table 7.17: The percentage of relevant and duplicated files for Yahoo search engine with and without using the proposed elimination scheme.

Keywords	Yahoo Without using the proposed elimination scheme		Percentage of relevance in the first 100 files of Yahoo query results	
	Number of results (Millions)	Percentage of duplicated files	Without using the proposed elimination scheme	With the proposed elimination scheme
Images				
Hotels	29,7	33%	86%	89%
Cars	59,8	34%	92%	95%
Dog	31,3	35%	87%	91%
Girls	41,6	36%	90%	93%
Weather	26,7	31%	82%	88%
Videos				
Videos	99,5	32%	91%	94%
Funny videos	12,8	31%	82%	85%
Dance movies	0.188	35%	80%	84%
Movie trailers	0.851	25%	80%	82%
Weight loss	0.242	29%	82%	84%
Audios				
Music	36,5	40%	84%	86%
Songs	2,25	37%	85%	86%
Albums	76,8	35%	79%	83%
MP3	0.77	36%	80%	85%
Clips	6,3	33%	81%	85%

Table 7.18: The percentage of relevant and duplicated files for Bing search engine with and without using the proposed elimination scheme.

Keywords	Bing Without using the proposed elimination scheme		Percentage of relevance in the first 100 files of Bing query results	
	Number of results (Millions)	Percentage of duplicated files	Without using the proposed elimination scheme	With the proposed elimination scheme
Images				
Hotels	24,1	30%	85%	88%
Cars	53,2	37%	95%	98%
Dog	28,2	39%	89%	95%
Girls	41,3	29%	93%	96%
Weather	24,3	33%	80%	85%
Videos				
Videos	111	33%	90%	93%
Funny videos	12,7	30%	85%	90%
Dance movies	0.185	36%	82%	86%
Movie trailers	0.911	27%	82%	84%
Weight loss	0.236	25%	83%	85%
Audios				
Music	34,6	37%	80%	83%
Songs	2,2	39%	83%	85%
Albums	63	30%	81%	84%
MP3	0.75	32%	87%	91%
Clips	5,2	31%	80%	84%

7.4 QBE Technique Performance

In this section, the performance of the enhanced QBE approach was studied in six experiments. The databases used in these experiments were the same as mentioned in Section 7.3. In the queries process, we have used the most frequently searched keywords and the most viewed multimedia files reported by [124, 125]. The server used in this study has the following characteristics: Intel Core i7 CPU, 4.0 GHz clock frequency, 16GB RAM, 256GB SSD, 1TB Hard Drive, and the operating system was Windows 7.

Experiment 1: Performance of clustering techniques in creating the databases

The aim of this experiment is to find out which clustering algorithm is more appropriate for the enhanced QBE. In this experiment, 100 million multimedia files have been clustered. Table 7.19 shows the percentage of the relevant files for some video queries when the following clustering algorithms are used in creating the database: C-means [113], subtractive [114], spectral [115], hierarchical [116], and neural network [117]. In addition, the Euclidean distance formula [83] has been used to find the similarity metrics of the clustered files. Furthermore, the percentage of the relevant files was found by two evaluators by checking the first 50 files of each query results. It is important to note that this and the second experiments were performed using the soft clustering technique, and the evaluators rank the queries results based on the following weight: a) 1 for most relevant files, b) 0.5 for approximately relevant files, and c) 0 for irrelevant files.

It is clear from Table 7.19 that these clustering algorithms have almost the same performance, and the percentage of the relevant files is between 60% and 70%. This percentage, however, does not satisfy user requirements.

Table 7.19: Percentage of relevant files for some video queries using K-means, subtractive, spectral, hierarchical, and neural network algorithms.

Title of uploaded video file	C-means	Subtractive	Spectral	Hierarchical	Neural Network
Gangnam Style	65.5%	68%	68.1%	66.3%	63.6%
Baby	67.5%	67%	68.1%	67.4%	64%
On The Floor	62.3%	64.1%	65.3%	62.9%	60.5%
Love The Way You Lie	61.9%	62.6%	67%	60.6%	61%
Party Rock Anthem	69%	65%	70.2%	62.6%	66.3%
Charlie Bit My Finger – Again!	66.5%	68.7%	69.8%	65.7%	64.9%
Waka Waka (This Time for Africa)	65%	65.9%	70%	69%	65%
Bad Romance	69%	70%	70.5%	69.7%	67.6%
Ai Se Eu Te Pego	67%	63.4%	68.2%	64%	67.1%
Danza Kuduro	64%	65%	70%	63.4%	63%

To improve the quality of the clustering results, we have built an ensemble system using k-means, subtractive and spectral, which will be used in the following experiments.

Experiment 2: The effects of specifying the number of clusters on the QBE performance

In this experiment, 100 million multimedia files have been clustered. Table 7.20 shows the effect of number of clusters on the percentage of relevant files for some video queries, where the percentage of the relevant files was found by two evaluators through checking the first 50 files of each query results. In this Table, the number of clusters used is 2000, 10000, 25000 and defined automatically as mentioned in

Chapter 6, Section 6.2.1. It is clear from this Table that using ensemble system increases the performance of the enhanced QBE, especially when the number of clusters is determined automatically, as the relevance was above 91% for this set of data.

Table 7.20: Effect of clusters number on the percentage of relevant files for some video queries using the proposed ensemble system.

Video file title	Number of clusters				
	2,000	5,000	10,000	25,000	Automatically defined
Gangnam Style	63.1%	65%	67.9%	85.3%	92%
Baby	66%	68%	69.5%	80.1%	91%
On The Floor	70.3%	71%	72.1%	82%	94.2%
Love The Way You Lie	71%	71.9%	72.5%	82.2%	93.1%
Party Rock Anthem	70.2%	71.7%	73%	84.4%	95%
Charlie Bit My Finger – Again!	72.8%	73.6%	74.5%	81%	92.9%
Waka Waka (This Time for Africa)	70%	71.9%	73.1%	82%	94.3%
Bad Romance	68.7%	70%	72.1%	83%	94.1%
Ai Se Eu Te Pego	66%	70.4%	71.7%	82.1%	92.5%
Danza Kuduro	68%	69.5%	70.5%	80%	91.6%

Based on the experimental work, it has been noticed that the hard clustering requires less processing time than the soft clustering, and therefore, it has been used in the following experiments. In these experiments, the evaluators were allowed to select a rank of 1 for the relevant files, and a rank of 0 for others, i.e., irrelevant files.

Experiment 3: Performance of sequential implementation versus clustering with and without parallel implementation

In this experiment, the performance of the sequential QBE scheme, the sequential QBE with clustering scheme and the proposed parallel QBE with clustering

implementation have been studied for different number of multimedia files. It is important to note the following: First, in the case of sequential QBE, the uploaded file is compared with all files in the database, while in the case of clustering, the uploaded file is compared with the leader of the clusters. Second, the execution time computed in this experiment represents the time required for comparing the uploaded file features with those of the database files. The result of this experiment is shown in Table 7.21. It's clear from this Table that the clustering techniques have improved the performance of QBE. In addition, the performance of the proposed QBE has been enhanced by using the parallel implementation.

Table 7.21: Execution time using the sequential QBE scheme, sequential QBE with clustering scheme and parallel QBE with clustering scheme for different number of multimedia files.

Number of multimedia files (Million)	Sequential QBE scheme	Sequential QBE with clustering scheme	The proposed QBE (Parallel with clustering implementation)	
	Time (Second)		# of threads	Time (Second)
0.5	109	3	8	0.12
0.75	147	8	12	0.19
1	204	11	16	0.22
10	509	15	16	0.45
100	812	19	16	0.89










Experiment 4: Performance of the enhanced QBE and Google QBE for image

The performance of the enhanced QBE is compared with that of the Google QBE for image queries, as Google QBE support images only. In this experiment, for every query, the number of the query results and the percentage of relevant files were recorded in Table 7.22. In this experiment, the number of Google results was provided by the search engine itself. In addition, the percentage of the relevant files

was found by two evaluators through checking the first 100 files of each query results. From Table 7.22, we can see the following:

- 1) The performance of the enhanced QBE outperforms the Google QBE. For instance, for this set of data, the percentage of relevant files was increased by a factor of 9% for the worst case and by a factor of 36% for the best case.
- 2) Although, the performance of Google QBE is acceptable for some files, it is found that it is weak for dealing with combined images and the images that have empty spaces such as the 6th and the 7th uploaded image shown in Table 7.22, respectively.


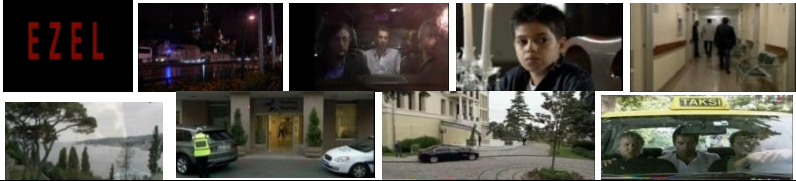



Table 7.22: Comparison between Google QBE and the enhanced QBE for image queries.

Uploaded file number	The uploaded image	The proposed QBE		Google QBE		Gain in percentage of relevant files
		Number of results	Percentage of relevant in first 100 files	Number of results	Percentage of relevant in first 100 files	
1		1150	90%	1200	65%	25%
2		1030	74%	1100	45%	29%
3		200	71%	150	35%	36%
4		1660	89%	1600	80%	9%
5		710	92%	650	70%	22%
6		540	71%	410	50%	21%
7		980	76%	1080	43%	33%
8		990	95%	865	70%	25%
9		1210	87%	1290	65%	22%

Experiment 5: The efficiency of the enhanced QBE approach for dealing with different video and audio files durations

In this experiment, the effect of the video/audio files length on the enhanced QBE performance was investigated. In this experiment, we have uploaded a group of multimedia files with different duration. Table 7.23 shows a sample of 5 queries from the tested group. It is important to note that the images shown in this Table represent the key frames of the video shots and they are used for the purpose of explanation. From the results shown in Table 7.23, we can see that the proposed approach is capable of dealing with video and audio files, and the percentage of the relevance was more than 91% for this set of data. Also it is clear from query #3 that although the number of the extracted key frames was small (3 key frames), the enhanced QBE was able to give more than 95% relevant files.

Table 7.23: The efficiency of the enhanced QBE approach for videos/audios.

Query #1	Key frames of the uploaded video			
				
User's options		Number of results (Thousand)	Required time (Seconds)	Percentage of relevant Files
a. Find exactly similar file.		0.97	3.3	100%
b. Find the full video.		0.66	1.87	98%
c. Find a high quality video.		0.45	1.73	98%
Query #2	Key frames of the uploaded video			
				
User's options		Number of results (Thousand)	Required time (Seconds)	Percentage of relevant Files
a. Find exactly similar file.		0.75	2.1	100%
b. Find the full video.		0.32	0.67	98%
c. Find a high quality video.		0.2	0.71	98%
Query #3	Key frames of the uploaded video			
				
User's options		Number of results (Thousand)	Required time (Seconds)	Percentage of relevant Files
a. Find exactly similar file.		2.57	2.2	100%
b. Find the full video.		4.91	3.67	91%
c. Find a high quality video.		5.07	3.71	95%
Query #4	Clips of the uploaded audio			
				
User's options		Number of results (Thousand)	Required time (Seconds)	Percentage of relevant Files
a. Find exactly similar file.		0.5	1.9	100%
b. Find the full audio.		0.25	1	93%
c. Find a high quality audio.		0.35	1.1	91%
Query #5	Clips of the uploaded audio			
				
User's options		Number of results (Thousand)	Required time (Seconds)	Percentage of relevant Files
a. Find exactly similar file.		1	1.75	100%
b. Find the full audio.		0.52	0.89	91%
c. Find a high quality audio.		0.28	0.59	90%

Experiment 6: Performance of the enhanced QBE versus the existing text based search engines: Google, Yahoo and Bing

Finally, we have done an experiment to study the overall performance of the enhanced QBE versus the existing text based search engines: Google, Yahoo and Bing. The average of the results of Google, Yahoo and Bing was compared with that of the enhanced QBE and shown in Table 7.24. It is important to note that the queries were performed using Google, Yahoo and Bing search engines by submitting the query keywords, i.e., title of multimedia file, while in the case of the enhanced QBE a video file was submitted as an example. It is clear from this Table that the enhanced QBE increased the percent of relevant files as compared with the average of Google, Yahoo and Bing by a factor of 6% in the worst case and a factor of 12% in the best case, for this set of data.

Table 7.24: Comparison between the Google, Yahoo and Bing text based search engines versus the enhanced QBE.

Multimedia type	Multimedia file title (Query Keywords)	Average of Google, Yahoo and Bing	Proposed QBE
		Percent of relevant files	
Images	Hotels	80%	91%
	Cars	88%	95%
	Dog	85%	94%
	Girls	89%	97%
	Weather	80%	92%
Videos	Gangnam Style	90%	96%
	Baby	80%	87%
	On The Floor	81%	87%
	Love The Way You Lie	80%	88%
	Party Rock Anthem	80%	89%
Audios	One Day	82%	89%
	Ai Se Eu Te	84%	92%
	Merry Go Round	80%	88%
	She Wolf (Falling To Pieces)	81%	89%
	Diamonds	80%	86%

Chapter 8

CONCLUSIONS AND FUTURE WORKS

This thesis provides solutions for some of the most common problems in the field of multimedia search engines. The main contributions of the thesis are:

- 1) Proposing a watcher based crawler (WBC).
- 2) Proposing a re-ranking approach.
- 3) Proposing an elimination approach for duplicated multimedia files.
- 4) Enhancing the QBE approach.

The advantages of these approaches are as follows: First, unlike the conventional crawlers, the developed watcher based crawler (WBC) has the ability of crawling both static and dynamic websites. In addition, WBC downloads the updated and the newly webpages only. Furthermore, it solves the crawlers overlapping and communication problems. Moreover, by splitting the proposed WBC into five units, where each unit is responsible for performing a specific crawling task, the overall crawling performance and the number of visited websites have been increased by a factor of approximately 3, 1.6, and 1.4 as compared with WEB-SAILOR [28], Apache Nutch [119], and Scrapy [120] crawlers, respectively.

Second, the developed re-ranking approach can efficiently deal with all multimedia files, and has the ability of showing the most relevant files to the top

of the query results, and increases the percentage of retrieved relevant files. In addition, the weight of the used descriptors is dynamically calculated and changed from a file to another based on the descriptors ability of distinguishing between the files. Furthermore, unlike most of the conventional re-ranking approaches, the proposed approach does not require any tuned parameters. Moreover, the re-ranking process does not require any explicit user intervention, and it works based on detecting some implicit user actions.

Thirdly, the developed approach which eliminate the duplicated files in multimedia query results has the following advantages 1) it can deal with any type of multimedia files like images, video and audio, 2) it can work with any search engine, and 3) it can also work in a routinely manner to deal with any updates on the databases. Moreover, by using the proposed elimination scheme, the comparison process will be done only once, and hence the query time will be decreased. In order to improve the efficiency of the proposed elimination scheme, parallel implementation is designed by making use of multiple processes.

Fourthly, the enhanced QBE performance was enhanced to support all multimedia types, to increase the percentage of relevant files and to decrease the process time in the query process. This is achieved through the following: First, a dynamic structure for clustering is used for creating and updating multimedia databases. Second, parallel implementation of the QBE has been designed. Finally, the enhanced QBE has the ability of eliminating duplicated multimedia files from the queries results by using the introduced elimination scheme.

As future work, the study presented in this thesis can be integrated with other types of search engines, like text based and topic specific search engines. For instance, the WBC can be integrated with VBScript and JavaFX RIAs techniques. In addition, WBC can be used to improve the performance of topic specific crawlers. Furthermore, the proposed re-ranking scheme can be used in the process of ranking and building the query results, instead of re-ranking the initial query outputs.

REFERENCES

- [1] Moschovitis, C. J., Poole, H., & Senft, T. M. (1999). *History of the Internet: A Chronology, 1843 to the Present*. AB C-CLIO, Incorporated.

- [2] Schwartz, C. (1998). Web search engines. *Journal of the American Society for Information Science*, Volume 49, Issue 11, Pages 973–982.

- [3] Agarwal, A., Singh, D., Pandey, A. K. A., & Goel, V. (2012). Design of a Parallel Migrating Web Crawler. *International Journal of Advanced Research in Computer Science and Software Engineering*, Volume 2, Issue 4, Pages 147-153.

- [4] Uzun, E., Agun, H. V., & Yerlikaya, T. (2013). A hybrid approach for extracting informative content from webpages. *Information Processing & Management*, Volume 49, Issue 4, Pages 928-944.

- [5] Amolochitis, E., Christou, I. T., Tan, Z. H., & Prasad, R. (2013). A heuristic hierarchical scheme for academic search and retrieval. *Information Processing & Management*, Volume 49, Issue 6, Pages 1326-1343.

- [6] Bruno, E. J. (2006). Ajax: asynchronous JavaScript and XML. *DR DOBBS JOURNAL*, Volume 31, Issue 2, Pages 32-35.

- [7] Cui, L. J., He, H., & Xuan, H. W. (2013). Analysis and Implementation of an Ajax-enabled Web Crawler. *International Journal of Future Generation Communication and Networking*, Volume 6, Issue 2, Pages 139-146.
- [8] Yao, Z., Daling, W., Shi, F., Yifei, Z., & Fangling, L. (2012). An Approach for Crawling Dynamic Webpages Based on Script Language Analysis. *Web Information Systems and Applications Conference (WISA)*, Pages 35-38.
- [9] Sharma, A. K., Gupta, J. P., & Agarwal, D. P. (2010). Parcahyd: an architecture of a parallel crawler based on augmented hypertext documents. *International Journal of Advancements in Technology*, Volume 1, Issue 2, Pages 270-283.
- [10] Zloof, M. M. (1975). Query by example, *National computer conference and exposition*. ACM, Pages 431-438.
- [11] Manning C., Raghavan P., & Schütze H. (2009). *Introduction to Information Retrieval*. Cambridge University Press, Online edition, Chapter 19, Pages 421-442.
- [12] Aybay I., & Alqaraleh S. (2010). Elimination of Repeated Occurrences in Image Search Engines. *9th International Conference on Application of Fuzzy Systems and Soft Computing*, Prague, Czech Republic, Pages 145-149.
- [13] Alqaraleh S. (2011). Elimination of Repeated Occurrences in Image Search Engines. *M.Sc. thesis*, Eastern Mediterranean University, Pages 31– 32.

- [14] Alqaraleh S., & Ramadan O. (2014). Elimination of Repeated Occurrences in Multimedia Search Engines. *The International Arab Journal of Information Technology*, Volume 11, No.2, Pages 134-139.
- [15] Alqaraleh S., & Ramadan O. (2015). Efficient Watcher Based Web Crawler. *Aslib Journal of Information Management*, Volume 67, Issue 6, Pages 663 – 686.
- [16] William, S., & Stallings, W. (2003). *Cryptography and Network Security: Principles and Practice*. 3/E, Prentice Hall, Page 681.
- [17] Manjunath, B. S., Salembier, P., & Sikora, T. (2002). *Introduction to MPEG-7: multimedia content description interface*. John Wiley & Sons, Volume 1.
- [18] Alqaraleh S., & Ramadan O. (2015). Utilizing Query by Example for Fast and Accurate Multimedia Retrieval. *Applied Mathematics & Information Sciences*, Volume 9, No. 1, Pages 125-134.
- [19] Alqaraleh S., & Ramadan O. (2012). Improving the performance of query by Sketch using parallel techniques. *3rd International Conference on Information and Communication Systems*, Jordan.
- [20] Langville, A. N., & Meyer, C. D. (2011). *Google's PageRank and beyond: The science of search engine rankings*. Princeton University Press.

- [21] Pichler, C., Holzmann, T., & Wright, B. (2011). Information Search and Retrieval. Crawler Approaches and Technology, Available at: http://www.iicm.tugraz.at/0x811bc82b_0x0011b4d6/ (accessed 9 September 2015), Pages 6-13.
- [22] Bhute, Avinash N., & Meshram, B. B. (2010). Intelligent Web Agent for Search Engines. *International Conference on Trends and Advances in Computational Engineering (TRACE - 2010)*, Pages 211-218.
- [23] Leng, A. G. K., Ravi, K. P., Singh, A. K., & Dash, R. K. (2011). PyBot: An Algorithm for Web Crawling. *International Conference on Nanoscience, Technology and Societal Implications (NSTSI -2011)*, IEEE, Pages 1-6.
- [24] Sharma, V., Kumar, M., & Vig, R. (2012). A Hybrid Revisit Policy for Web Search. *Journal of Advances in Information Technology*, Volume 3, Issue 1, Pages 36-47.
- [25] Singh, A. V., & Vikas, A. M. (2014). A Review of Web Crawler Algorithms. *International Journal of Computer Science & Information Technologies*, Volume 5, Issue 5, Pages 6689-6691.
- [26] Cho, J., Garcia-Molina, H., & Page, L. (2012). Reprint of: Efficient crawling through URL ordering. *Computer Networks*, Volume 56, Issue 18, Pages 3849-3858.

- [27] Olston, C., & Najork, M. (2010). Web crawling. *Foundations and Trends in Information Retrieval*, Volume 4, Issue 3, Pages 175-246.
- [28] Mukhopadhyay, D., Mukherjee, S., Ghosh, S., Kar, S., & Kim, Y. C. (2011). Architecture of a Scalable Dynamic Parallel WebCrawler with High Speed Downloadable Capability for a Web Search Engine. *6th International Workshop on MSPT*, Pages 103-108.
- [29] Markatos, E. P. (2001). On caching search engine query results. *Computer Communications*, Volume 24, Issue 2, Pages 137-143.
- [30] Yang, Y., Xu, D., Nie, F., Luo, J., & Zhuang, Y. (2009). Ranking with local regression and global alignment for cross media retrieval. *The 17th ACM international conference on Multimedia*, Pages 175-184.
- [31] Dali, L., Fortuna, B., Duc, T. T., & Mladenić, D. (2012). Query-independent learning to rank for rdf entity search The Semantic Web: Research and Applications. *Springer Berlin Heidelberg*, Pages 484-498.
- [32] Thollard, F., & Quénot, G. (2013). Content-based re-ranking of text-based image search results. *Advances in Information Retrieval*, Springer Berlin Heidelberg, Volume 7814, Pages 618-629.
- [33] Liu, & Tie-Yan. (2011). Query-Dependent Ranking. Learning to Rank for Information Retrieval. *Springer Berlin Heidelberg*, Pages 113-121.

- [34] Jain, V., & Varma, M. (2011). Learning to re-rank: query-dependent image re-ranking using click data. *20th international conference on World Wide Web*, Pages 277-286.
- [35] Zhu, Y., Xiong, N., Park, J., & He, R. (2008). A Web Image Retrieval Re-ranking Scheme with Cross-Modal Association Rules. *International Symposium on Ubiquitous Multimedia Computing*, Issue 13, Pages 83 - 86.
- [36] Bellini, P., Cenni, D., & Nesi, P. (2012). On the Effectiveness and Optimization of Information Retrieval for Cross Media Content. *In KDIR*, Pages 344-347.
- [37] Li, L., Zhai, X., & Peng, Y. (2012). Tri-space and ranking based heterogeneous similarity measure for cross-media retrieval. *21st International Conference on Pattern Recognition (ICPR)*, IEEE.
- [38] Bo, L., Wang, G., & Yuan, Y. (2012). A novel approach towards large scale cross-media retrieval. *Journal of Computer Science and Technology*, Volume 27, Issue 6, Pages 1140-1149.
- [39] Mao, X., Lin, B., Cai, D., He, X., & Pei, J. (2013). Parallel field alignment for cross media retrieval. *In Proceedings of the 21st ACM international conference on Multimedia*, Pages 897-906.
- [40] Kang, C., Liao, S., He, Y., Wang, J., Xiang, S., & Pan, C. (2014). Cross-Modal Similarity Learning: A Low Rank Bilinear Formulation. arXiv preprint arXiv:1411.4738.

- [41] Safadi, B., & Quénot, G. (2011). Re-ranking by local re-scoring for video indexing and retrieval. *In Proceedings of the 20th ACM international conference on Information and knowledge management*, Pages 2081-2084.
- [42] Jin, H., He, R., Liao, Z., Tao, W., & Zhang, Q. (2006). A Flexible and Extensible Framework for Web Image Retrieval System. *International Conference on Internet and Web Applications and Services/Advanced*, French, Pages 193 – 198.
- [43] Doulaverakis, H., Nidelkou, E., Gounaris, A., & Kompatsiaris, Y. (2006). A Hybrid Ontology and Content-Based Search Engine for Multimedia Retrieval. *ADBIS Research Communications*.
- [44] Mirzal, A. (2012). Design and Implementation of a Simple Web Search Engine. *International Journal of Multimedia and Ubiquitous Engineering*, Volume 7, No. 1, Pages 53-60.
- [45] Maredj, A. E., & Tonkin, N. (2011). Extending the Conceptual Neighborhood Graph of the Relations for the Semantic Adaptation of Multimedia Documents. *International Conference on Education and Information Technology*, Volume 53, Pages 80-83.
- [46] YuJie, L., Feng, B., ZongMin, L., & Hua, L. (2013). 3D Model Retrieval Based on 3D Fractional Fourier Transform. *The International Arab Journal of Information Technology*, Online Publication, Volume 10, No.5.

- [47]Ding, Z., Dai, J., Gao, X., & Yang, Q. (2012). A Hybrid Search Engine Framework for the Internet of Things. *Ninth Web Information Systems and Applications Conference*, Pages 57 - 60.
- [48]Chau, M., & Wong, C. H. (2010). Designing the user interface and functions of a search engine development tool. *Decision Support Systems*, Volume 48, Issue 2, Pages 369-382.
- [49]Netcraft (2014), “Web server survey”, Available at: <http://www.news.netcraft.com/archives/2014/01/03/january-2014-web-server-survey.html/> (accessed 4 July 2014).
- [50]Kumar, M. S., & Neelima, P. (2011). Design and Implementation of Scalable, Fully Distributed Web Crawler for a Web Search Engine. *International Journal of Computer Applications*, Volume 15, Issue 7, Pages 8-13.
- [51]Wu, Min, & Lai, J. (2010). The Research and Implementation of parallel web crawler in cluster. *Computational and Information Sciences (ICCIS)*, Pages 704-708.
- [52]Amin, Ruhul, M., Prince, M. A., & Hussain, M. A. (2012). WEBTracker: A Web Crawler for Maximizing Bandwidth Utilization. *Journal of Science and Technology*, Volume 16, No.2, Pages 32-40.

- [53] Yang, Y., Du, Y., Hai, Y., & Gao, Z. (2009). A Topic-Specific Web Crawler with Web Page Hierarchy Based on HTML Dom-Tree. *Asia-Pacific Conference on Information Processing*, Pages 420 - 423.
- [54] Mukhopadhyay, D., & Sinha, S. (2008). A New Approach to Design Graph Based Search Engine for Multiple Domains Using Different Ontologies. *International Conference on Information Technology (ICIT '08)*, Pages 267 - 272.
- [55] Dincturk, M. E., Jourdan, G. V., Bochmann, G. V., & Onut, I. V. (2014). A model-based approach for crawling rich internet applications. *ACM Transactions on the Web (TWEB)*, Volume 8, Issue 3, Pages 1-19.
- [56] Cui, L. J., He, H., Xuan, H. W., & Li, J. G. (2013). Design and Development of an Ajax Web Crawler. *Tenth International Conference on Computability and Complexity in Analysis*, Volume 17, Pages 6-10.
- [57] Nath, R., & Bal, S. (2011). A novel mobile crawler system based on filtering off non-modified pages for reducing load on the network. *The International Arab Journal of Information Technology*, Volume 8, Issue 3, Pages 272-279.
- [58] Mishra, S., Jain, A., & Sachan, A. K. (2010). Smart Approach to Reduce the Web Crawling Traffic of Existing System using HTML based Update File at Web Server. *International Journal of Computer Applications*, Volume 11, Issue 7, Pages 34-38.

- [59] Bhushan, B., Gupta, M., & Gupta, G. (2012). Increasing the Efficiency of Crawler Using Customized Sitemap. *International Journal of Computing and Business Research*, Online Publication, Volume 3, Issue 2.
- [60] Schonfeld, U., & Shivakumar, N. (2009). Sitemaps: above and beyond the crawl of duty. *18th international conference on World Wide Web*, ACM, Pages 991-1000.
- [61] Brawer, S. B., Ibel, M., Keller, R. M., & Shivakumar, N. (2013). Web crawler scheduler that utilizes sitemaps from websites. *U.S. Patent Application* 13/858,872, Available at: <http://www.google.com/patents/US7769742/> (accessed 9 September 2015).
- [62] Duda, C., Frey, G., Kossmann, D., Matter, R., & Zhou, C. (2009). Ajax crawl: Making Ajax applications searchable. *25th IEEE International Conference on Data Engineering*, IEEE, Pages 78-89.
- [63] Chen, Y., Sanghavi, S., & Xu, H. (2012). Clustering sparse graphs. *In Advances in Neural Information Processing Systems 25 (NIPS 2012)*, Pages 2204-2212.
- [64] Pan, J., Hou, Y. T., & Li, B. (2003). An overview of DNS-based server selections in content distribution networks. *Computer Networks*, Volume 43, Issue 6, Pages 695-711.

- [65] Kausar, M. A., Dhaka, V. S., & Singh, S. K. (2013). Web Crawler – A Review. *International Journal of Advanced Research in Computer Science and Software Engineering*, Volume 3, Issue 8, Pages 31-36.
- [66] Choudhary, S., Dincturk, M. E., Mirtaheri, S. M., Moosavi, A., von Bochmann, G., Jourdan, G. V., & Onut, I. V. (2012). Crawling rich internet applications: the state of the art. *Conference of the Center for Advanced Studies on Collaborative Research (CASCON 2012)*, Pages 146-160.
- [67] V8 (2014). V8 3.31.1. Available at: <https://www.developers.google.com/v8/> (accessed 7 April 2014).
- [68] Chromium (2014). Chromium. Available at: <https://www.chromium.org/Home/> (accessed 7 April 2014).
- [69] Ward, E., & French, G. (2013). *Ultimate Guide to Link Building: How to Build Backlinks. Authority and Credibility for Your Website, and Increase Click Traffic and Search Ranking*, Chapter 9, Entrepreneur Press.
- [70] Kelly, B., & Nixon, W. (2013). SEO analysis of institutional repositories: What's the back story?. *Open Repositories (2013)*, Canada.
- [71] CHEN, C., LIN, L., & SHYU, M. (2012). Re-Ranking Algorithm for Multimedia Retrieval via Utilization of Inclusive and Exclusive Relationships between Semantic Concepts. *International Journal of Semantic Computing*, Volume 06, Issue 02, Pages 135-154.

- [72]Kang, C., Wang, X., Chen, J., Liao, C., Chang, Y., Tseng, B., & Zheng, Z. (2011). Learning to re-rank web search results with multiple pairwise features. *Proceedings of the fourth ACM international conference on Web search and data mining*, Pages 735-744.
- [73]Mythili, C., & Kavitha, V. (2011). Efficient Technique for Color Image Noise Reduction. *The research bulletin of Jordan, ACM*, Volume 1, Issue 11, Pages 41-44.
- [74]Bhattacharyya, S. (2011). A brief survey of color image preprocessing and segmentation techniques. *Journal of Pattern Recognition Research*, Volume 1, Issue 1, Pages 120-129.
- [75]Chenglong, C., & Ni, J. (2012). Median filtering detection using edge based prediction matrix, *Digital Forensics and Watermarking*. Springer Berlin Heidelberg, Pages 361-375.
- [76]Image Deskewing (2014). Document Skew Checker. Available at: <http://www.aforgenet.com> (accessed 7 April 2014).
- [77]Naveed, E., Bin Tariq, T., & Baik, S. (2012). Adaptive key frame extraction for video summarization using an aggregation mechanism. *Journal of Visual Communication and Image Representation*, Volume 23, Issue 7, Pages 1031-1040.

- [78] Kalogeiton, S., Papadopoulos, P., Chatzichristofis, S., & Boutalis, Y. (2010). A Novel Video Summarization Method Based on Compact Composite Descriptors and Fuzzy Classifier. *1st International Conference for Undergraduate and Postgraduate Students in Computer Engineering, Informatics, related Technologies and Applications*, Pages 237-246.
- [79] Rajendra, P., & Keshaveni, N. (2014). A Survey of Automatic Video Summarization Techniques. *International Journal of Electronics, Electrical and Computational System*, Online Publication, Volume 2.
- [80] Duplicate Songs Detector via Audio Fingerprinting (2013). Available at: <http://www.codeproject.com/Articles/206507/Duplicates-detector-via-audio-fingerprinting> (accessed 7 April 2014).
- [81] Baluja, S., & Covell, M. (2006). Content Fingerprinting Using Wavelets. *European Conference on Visual Media Production (CVMP)*, Pages 198-207.
- [82] Bribiesca, E. (1999). A new chain code. *Pattern Recognition*, Volume 32, Issue 2, Pages 235-251.
- [83] Danielsson, P. E. (1980). Euclidean distance mapping. *Computer Graphics and image processing*. Volume 14, Issue 3, Pages 227-248.
- [84] Rao, R., & Yip, P. (2014). *Discrete cosine transform: algorithms, advantages, applications*. Chapter 7, Academic press, Pages 136-163.

- [85]Lux, M., & Chatzichristofis, S. (2008). Lire: Lucene Image Retrieval – An Extensible Java CBIR Library. *ACM International Conference on Multimedia (ACM MM)*, British, Pages 1085-1087.
- [86]Balan, S., & Devi, T. (2012). Design and Development of an Algorithm for Image Clustering In Textile Image Retrieval Using Color Descriptors. *International Journal of Computer Science, Engineering and Applications (IJCSEA)*, Volume 2, Issue 3, Pages 199-211.
- [87]Chatzichristofis, S., & Boutalis, Y. (2008). CEDD: Color and Edge Directivity Descriptor: A Compact Descriptor for Image Indexing and Retrieval. 6th *International Conference in advanced research on Computer Vision Systems (ICVS)*, Greece, Pages 312–322.
- [88]Chatzichristofis, S., & Boutalis, Y. (2008). FCTH: Fuzzy Color and Texture Histogram - A Low Level Feature for Accurate Image Retrieval. 9th *International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*, IEEE Computer Society, Austria, Pages 191 – 196.
- [89]Chatzichristofis, S., Boutalis, Y., & Lux, M. (2011). Combining Color and Spatial Color Distribution Information in a Fuzzy Rule Based Compact Composite Descriptor. *Communications in Computer and Information Science*, Springer-Verlag, Pages 49-60.

- [90]Kekre, H., Bhandari, N., Nair, N., Padmanabhan, P., & Bhandari, S. (2013). A Review of Audio Fingerprinting and Comparison of Algorithms. *International Journal of Computer Applications*, Volume 70, Issue 13, Pages 24-30.
- [91]Baeza-Yates, R., Hurtado, C., Mendoza, M., & Dupret, G. (2005). Modeling user search behavior. *Third Latin American Web Congress (LA-WEB)*, IEEE, Pages 242 – 251.
- [92]Ngo, C. W., Xu, C., Kraaij, W., & El Saddik, A. (2013). Web-Scale Near-Duplicate Search: Techniques and Applications. *MultiMedia*, IEEE, Volume 20, Issue 3, Pages 10-12.
- [93]Chiu, C. Y., Tsai, T. H., & Hsieh, C. Y. (2012). Scalable near-duplicate video stream monitoring. *In Intelligent Signal Processing and Communications Systems (ISPACS)*, Pages 12-15.
- [94]Wu, X., Hauptmann, A. G., & Ngo, C. W. (2007). Practical Elimination of Near-Duplicates from Web Video Search. *ACM International Conference on Multimedia*, Pages 218-227.
- [95]Wu, X., Ngo, C. W., Hauptmann, A. G., & Tan, H. K. (2009). Real-Time Near-Duplicate Elimination for Web Video Search with Content and Context. *IEEE Transactions on Multimedia*, Volume 11, Issue 2, Pages 196 - 207.

- [96] Dong, W., Wang, Z., Charikar, M., & Li, K. (2012). High-confidence near-duplicate image detection. *2nd ACM International Conference on Multimedia Retrieval*, Pages 1-8.
- [97] Cheddad, A., Condell, J., Curran, K., & McKevitt, P. (2010). A hash-based image encryption algorithm. *Optics Communications*, Volume 283, Issue 6, Pages 879-893.
- [98] Kasutani, E., & Yamada. (2001). The MPEG-7 color layout descriptor: a compact image feature description for high-speed image/video segment retrieval. *International Conference on Image Processing*, IEEE, Volume 1, Pages 674-677.
- [99] Ro, Y. M., Kim, M., Kang, H. K., Manjunath, B. S., & Kim, J. (2001). MPEG-7 homogeneous texture descriptor. *ETRI journal*, Volume 23, Issue 2, Pages 41-51.
- [100] Latecki, L. J., Lakämper, R., & Eckhardt, U. (2000). Shape descriptors for non-rigid shapes with a single closed contour. *IEEE Conference on Computer Vision and Pattern Recognition*, Volume 1, Pages 424-429.
- [101] Dimitriadou, K., Papaemmanouil, O., & Diao, Y. (2014). Explore-by-example: An automatic query steering framework for interactive data exploration. *The ACM SIGMOD international conference on Management of data*, Pages 517-528.

- [102] Watai, Y., Yamasaki, T., & Aizawa, K. (2007). View-Based Web Page Retrieval using Interactive Sketch Query. *International Conference on Image Processing*, Volume 6, Pages 357 – 360.
- [103] Giangreco, I., Springmann, M., Al Kabary, I., & Schuldt, H. (2012). A User Interface for Query-by-Sketch Based Image Retrieval with Colour Sketches. *Computer Science*, Volume 7224, Pages 571-572.
- [104] Wang, Y., Yu, M., Jia, Q., & Guo, H. (2011). Query by sketch: An asymmetric sketch-vs-image retrieval system. *Image and Signal Processing (CISP)*, Volume: 3, Pages 1368 – 1372.
- [105] Yoon, S. M., & Kuijper, A. (2010). Query-by-sketch based image retrieval using diffusion tensor fields. *Image Processing Theory Tools and Applications (IPTA)*, Pages 343 - 348.
- [106] Vermilyer, R. (2006). Intelligent User Interface Agents in Content-Based Image Retrieval. *SoutheastCon*, IEEE, New York, Pages 136-142.
- [107] Shirahama, K., & Uehara, K. (2011). Utilizing Video Ontology for Fast and Accurate Query-by-Example Retrieval. *Fifth IEEE International Conference on Semantic Computing (ICSC)*, Pages 395 – 402.
- [108] Gao, X., Li, X., Feng, J., & Tao, D. (2009). Shot-based video retrieval with optical flow tensor and HMMs. *Pattern Recognition Letters*, Volume 30, Issue 2, Pages 140–147.

- [109] Shen, H. T., Shao, J., Huang, Z., & Zhou, X. (2009). Effective and Efficient Query Processing for Video Subsequence Identification. *IEEE transactions on knowledge and data engineering*, Volume 21, NO. 3, Pages 321-334.
- [110] Helén, M., & Virtanen, T. (2010). Audio query by example using similarity measures between probability density functions of features. *EURASIP Journal on Audio, Speech, and Music Processing*, Volume 2010, Pages 1-12.
- [111] Chen, L. C. (2011). Using a new relational concept to improve the clustering performance of search engines. *Information Processing and Management*, Volume 47, Issue 2, Pages 287-299.
- [112] Hindle, A., Shao, J., Lin, D., Lu, J., & Zhang, R. (2011). Clustering Web video search results based on integration of multiple features. *World Wide Web*, Volume 14, Issue 1, Pages 53-73.
- [113] Chen, W. C., & Wang, M. S. (2009). A fuzzy c-means clustering-based fragile watermarking scheme for image authentication. *Expert Systems with Applications*, Volume 36, Issue 2, Part 1, Pages 1300–1307.
- [114] Alnihoud, J. (2011). Image Retrieval System with Self Organizing Map and Subtractive Fuzzy Clustering. *International Journal on Information and Communication Technologies*, Volume 4, Issue 3-4, Pages 103-110.

- [115] Tung, F., Wong, A., & Clausi, D. A. (2010). Enabling scalable spectral clustering for image segmentation. *Pattern Recognition*, Volume 43, Issue 12, Pages 4069–4076.
- [116] Murthy, V. S. V. S., Vamsidhar, E., Kumar, J. S., & Rao, P. S. (2010). Content Based Image Retrieval using Hierarchical and K-Means Clustering Techniques. *International Journal of Engineering Science and Technology*, Volume 2, Issue 3, Pages 209-212.
- [117] Sowmya, B., & Rani, B. S. (2011). Colour image segmentation using fuzzy clustering techniques and competitive neural network. *Applied Soft Computing*, Volume 11, Issue 3, Pages 3170–3178.
- [118] Woźniak, M., Graña, M., & Corchado, E. (2014). A survey of multiple classifier systems as hybrid systems. *Information Fusion*, Volume 16, Pages 3-17.
- [119] Apache Nutch (2015). Nutch 2.3. Available at: <http://www.nutch.apache.org> (accessed 20 May 2015).
- [120] Scrapy (2015). Scrapy 1.0. Available at: <http://www.scrapy.org> (accessed 20 May 2015).
- [121] Statistics and Social Network of YouTube Videos (2008). Available at: <http://netsg.cs.sfu.ca/youtubedata> (accessed 20 May 2015).

- [122] Cheng, X., Dale, C., & Liu, J. (2008). Statistics and social network of YouTube videos. *16th international workshop on quality of service*, IEEE, Pages 229-238.
- [123] Craswell, N. (2009). *R-Precision*, *Encyclopedia of Database Systems*. Springer US, Pages 2453-2453.
- [124] Multimedia Search keywords (2014). Available at: <https://www.wordtracker.com> (accessed 20 May 2015).
- [125] Search keywords (2014). Available at: <http://www.wordstream.com> (accessed 20 May 2015).