



Libyan Academy – Misurata
School of Engineering and
Applied Science
Department of Information
Technology

Using Web Pages Dynamicity to Prioritise
Web Crawling

A Thesis Submitted in Partial Fulfillment of the
Requirements For The Master Degree in Information Technology

By:

Nisreen Mahmoud Alderratia

Supervised by:

Dr. Mohammed Mosbah Elsheh

2019

Acknowledgment

I would like to express my very great appreciation to the precious soul of my mother whom I have always had the feeling of having her by my side and that she never left for even a moment. Thank you for providing me with the opportunity to be where I am today.

I am particularly grateful to my sisters: Fatima, for being a mother, for dedicating her time to help me and for her ultimate love and support. Sharifa, for being a candle that lights my path and no words can truly express what she has done for me. Nawara, who has taught me the meaning of kindness and compassion, and for putting up with me through the toughest moments of my life. Thank you all sincerely for providing me with endless inspiration to pursue my scientific and practical career. You are indeed the secret of my happiness.

My special thanks are extended to Dr.Mohammed Elsheh whom I am indebted in that he has taught and supported me more than I could ever give him credit for. I do not know how to thank him enough. May Allah reward him well.

Abstract

Web crawling is a process performed by web crawlers. In fact, it is considered the main process, which is used by search engines to collect web pages from the web, in order to be indexed and used for displaying the search results according to users' requirements. In addition, web crawlers must continually revisit Web pages, to keep the search engine database updated, since the web pages are constantly uploaded and modified. Moreover, it is fundamental to determine in the crawling process, the most important pages to be recrawled first. This is to avoid the time limitation and network issues that face the web crawling process. Thus, this study attempts to introduce a method that is used to indicate the crawler, specifically, in order to identify in what order it should recrawl web pages that have been crawled before, as to acquire more important and valuable pages earlier than others. In addition, the researcher proposed a web crawling strategy which is based on the topic similarity, accompanied with the dynamicity of web pages, where the crawler was downloading relevant pages and recrawling them recursively. Also, every time a change emerged in one of the pages, its counter increased. Therefore, if the page was relevant and changed frequently it would be considered an important page and was given a high priority in the crawling process. The dissertation results indicated that using web pages' dynamicity is an effective way for prioritising web pages in the crawling process, in order to obtain the highest dynamic pages first, as there is a high possibility of being changed in terms of their content, before the least dynamic ones.

Table of Contents

Acknowledgement	2
Abstract	3
List of Tables	7
Table of Figures.....	8
List of Abbreviations.....	10
Chapter One	11
Introduction	11
1.1 Problem Statemnt	13
1.2 Goal and Objectives	13
1.3 Scope of the Study	14
1.4 Thesis Outline	14
1.5 Summary	15
Chapter Two	16
Litrature Review	16
2.1 General Web Crawler	16
2.2 Focused Web Crawler	18
2.3 Detecting changes in web pages	20
2.4 Prioritising URLs in the Frontier	22
2.5 Summary	24
Chapter Three	26
Methodology.....	26
3.1 Focused web crawler	27
3.1.1 Terms Selection	27
3.1.2 Determining Similarity Threshold Value	29

3.1.3 Fetching Web Pages	29
3.1.4 Parsing Web Pages	30
3.1.5 Classifying Web Pages	30
3.1.6 Reading Robot File	32
3.1.7 Extracting URLs of Web Pages.....	33
3.1.8 Updating Web Pages Repository	34
3.2 Revisiting and Detecting Changes in Web Pages	34
3.2.1 Detecting Changes of Web Pages.....	35
3.2.1.1 Detecting Structural Changes	36
3.2.1.2 Detecting Textual Changes	37
3.3 Priority of Web Pages in the Frontier (Ordering Schema)	39
3.4 Evaluation Phase	40
3.5 Summary	40
Chapter Four	42
Implementation	42
4.1 Data structure	42
4.2 Classes	43
4.2.1 Class 1: Fetching Web Pages	44
4.2.2 Class 2: Parsing Web Pages	45
4.2.3 Class 3 : Terms Selection	45
4.2.4 Class 4: Determining Similarity Threshold Value	47
4.2.5 Class 5: Classifying Web Pages	49
4.2.6 Class 6: Reading Robot File	52
4.2.7 Class 7 : Extracting URLs of Web Pages.....	54
4.2.8 Class 8: Detecting Changes of Web Pages.....	57
4.2.8.1 Structural Changes	58
4.2.8.2 Textual changes	59

4.2.9 Class 9: The Frontier	63
4.3 Summary	67
Chapter Five	68
Results and Discussions	68
5.1 News Terms	68
5.2 Maximum Number of URLs for Websites	68
5.3 Categories Characteristics	69
5.3.1 Number of Pages for Each Category	69
5.3.2 Focused Crawler Performance for Each Category	70
5.3.3 Changes of News Categories Web Pages	73
5.3.3.1 Re-Categorising Categories Web Pages	73
5.3.3.2 Structural and Textual Changes of Categories Web pages	74
5.4 The Frontier	76
5.5 Dynamicity approach versus (<i>PageRank, Backlink Count and Forward Link Count</i>)	80
5.6 Summary	82
Chapter Six	83
Conclusion and Future Works	83
6.1 Conclusion	83
6.2 Future Works	83
References	85
Appendix A	88
Appendix B.....	89
Appendix C.....	91
Appendix D.....	95
Appendix E.....	98
Appendix F.....	101
Appendix G.....	103

List of Tables

Table 5.1	Number of URLs for Each News Category	69
Table 5.2	Priority Order of Top Five Most Dynamic Page in Business Category Compared with PageRank, Backlink Count and Forward link Count Algorithms	80
Table 5.3	Priority Order of Top Five Most Dynamic Page in Crime Category Compared with PageRank, Backlink Count and Forward link Count Algorithms	80
Table 5.4	Priority Order of Top Five Most Dynamic Page in Health Category Compared with PageRank, Backlink Count and Forward link Count Algorithms	81
Table 5.5	Priority Order of Top Five Most Dynamic Page in Political Category Compared with PageRank, Backlink Count and Forward link Count Algorithms	81
Table 5.6	Priority Order of Top Five Most Dynamic Page in Sports Category Compared with PageRank, Backlink Count and Forward link Count Algorithms	81
Table 5.7	Priority Order of Top Five Most Dynamic Page in Technology Category Compared with PageRank, Backlink Count and Forward link Count Algorithms	81

List of Figures

Figure 1.1 Search Engine System Architecture (Dutta & Bansal, 2016)	11
Figure 2.1 General Architecture of Web Crawler (Udapure et al., 2014)	17
Figure 3.1 Overall Crawling Process	27
Figure 3.2 Types of Structural Changes in Web Pages	36
Figure 3.3 Show How to Create and Detect Changes in Textual Dynamic Parts of Web Pages.....	38
Figure 4.1 Database Schema	43
Figure 4.2 The Main Operation Steps of the Proposed Web Crawler	44
Figure 4.3 Neglected Relevant URLs	46
Figure 4.4 The Most Frequent Top Ten Terms from URLs in Table (2)	47
Figure 4.5 Similarity Threshold for Eighth Crawling Process	48
Figure 4.6 Dot Product Between Web Page Vector and News Categories Vectors	51
Figure 4.7 Results of Computing Cosine Similarity of (http://www.bbc.com/sport/tennis) Web Page.....	52
Figure 4.8 A Portion of (www.bbc.com) Robot File Text String.....	53
Figure 4.9 Separating the Line of the Robot File string.....	53
Figure 4.10 Absolute URLs Extracted of (http://www.bbc.com/sport/tennis) Web Page	55
Figure 4.11 Root Relative URLs Extracted of (http://www.bbc.com/sport/tennis) Web Page.....	56
Figure 4.12 Normalising Dot-Dot-Slash ("../") URLs.....	56
Figure 4.13 Recategorising Web Pages	58
Figure 4.14 Web pages of Business Category had Structurally Changed	59
Figure 4.15 Textual Changes in (http://www.bbc.com/news/correspondents/davelee) Web Page...	60
Figure 4.16 The Process of Handling Path of Text Node	60
Figure 4.17 Paths of #Text[1] Nodes	61
Figure 4.18: Changes in Textual Dynamic Nodes of (http://www.bbc.com/news/correspondents/davelee) Web Page.....	62

Figure 4.19 Textual Dynamic Part in (http://www.bbc.com/news/correspondents/davelee) Web Page.....	63
Figure 4.20 The Results of Crawling and Classifying Seed URLs Pages	64
Figure 4.21 Examples of Crawled URLs in First Cralwing Process.....	65
Figure 5.1 Number of Reletive Web Pages and Harvest Ratio For All News Categories	72
Figure 5.2 Percentage of Web Pages Their Category Has Changed Through The seventh Crawling Phases.....	73
Figure 5.3 Total of Structural and Textual Changes Through Crawling Phases.....	74
Figure 5.4 Total Changes Appeared in Categories Web Pages	75
Figure 5.5 Dynamic and Static URLs For All Categories Through All Crawling Phases	77
Figure 5.6 Portions of Dynamic and Static URLs at First, Middle and Final Crawling Phases	79
Figure 5.7 Dynamic Parts of (http://www.aljazeera.com/news/2017/01/donald-trump-russian-hacking-election-outcome-170107034647205.html)	79

List of Abbreviations

DOM	Document Object Model
HMM	Hidden Markov Model
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
I	Importance
IB	Backlink Importance
IC	Combined Importance
ID	Dynamicity Importance
IDF	Inverse Document Frequency
IF	Forward link Importance
IR	PageRank Importance
IS	Similarity Importance
LS	Link Score
P	Page
SSRM	Semantic Similarity Retrieval Model
TF	Term Frequency
URL	Uniform Resource Locator
VSM	Vector Space Model

Chapter One

Introduction

Internet has become an operative mean for sharing information among its users around the globe. Through documents called web pages commonly written in HTML (Hyper Text Markup Language) and has a unique identifier called uniform resource locator (URL), it can successfully provide information. Additionally, in order to make users updated with new information that serve their needs, the process of uploading new pages has to happen constantly .

Generally, search engines are computer programs, which gather and organise web pages, in order to be retrieved in later stages, depending on a user's query. In addition, the search engines are commonly made up of three main components. The first is the web crawler to collect web pages from the web and store them in search engine's database; the second is the indexer to give a brief overview of web pages by extracting the keywords from their content, and the last one is the user interface, which is an interface where users can write through their queries. The system of work and interaction between these components demonstrates the methodology of execution inside the search engines (Dutta and Bansal, 2016), is shown in Figure 1.1.

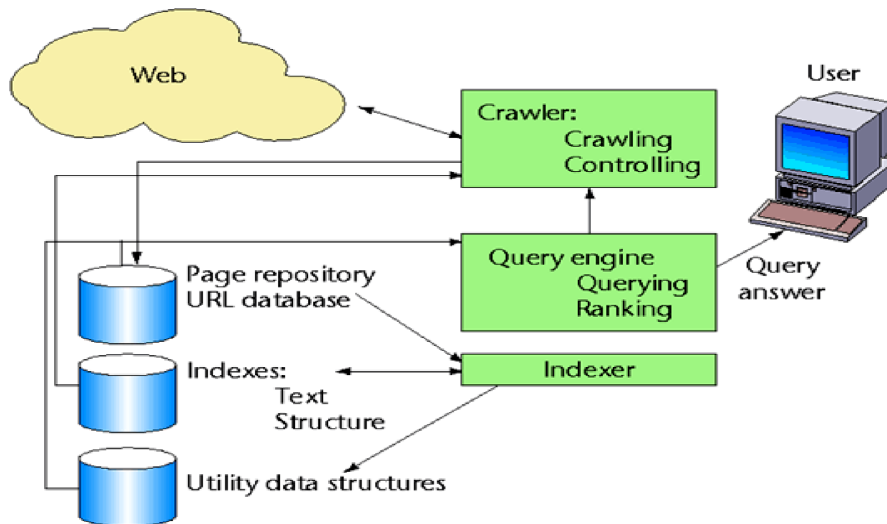


Figure 1.1: Search Engine System Architecture (Dutta and Bansal, 2016)

Web pages have to be crawled and aggregated by a program called the crawler in order to be delivered to search engines so those web pages can be indexed and summarised. Additionally, the crawler begins crawling through using an initial URL that is added to a queue, which is known as

the frontier. Then its web page is downloaded, and its content is parsed in order to extract its URLs and add them to the frontier in a specific order, where they can download and parse their web pages later on. This crawling process of extracting URLs from the frontier, which is downloading and parsing their pages, continues for either a certain number of URLs or depending on network resources. However, the crawler performance is generally evaluated by its efficiency to retrieve valid and valuable web pages for a search engine.

(Cho, Garcia-Molina and Page, 1998) mentioned that in order to build a good crawler, two difficulties demand to be addressed. First, the crawler must avert excess in downloading the web pages during the crawling, so it can only load a specific number of pages, even though the network contains a huge number of pages to load. Moreover, due to the limitation of time and resources, the crawler should accurately determine which pages should be crawled first and the arrangement in which they are crawled. Second, because of the dynamic nature of web pages that change from time to time, when the crawler crawls and stores web pages in the search engine's database, they become outdated compared to web-based versions. Therefore, the crawler has to determine regular intervals in which it revisits web pages in seek of keeping the search engine's database up to date.

Due to the rapid growth of websites and the increasing number of web users around the world, general crawlers become incompetent to crawl and handle this huge number of web pages. This has led to urgent need to design a new type of web crawlers that can search for the required information efficiently and accurately. This kind of web crawlers is called focused web crawler as it crawl pages that are relevant to a particular subject while neglecting the irrelevant ones. However, (Batsakis, Petrakis and Milios, 2009), stated that this process leads to minimising the number of resources, such as: time, space and network bandwidth, with regard to accomplish the search task.

As in the case of web crawlers, web pages vary in their importance from page to page. Some of them are of a high importance while others are considered less important. For this reason, when crawling web pages, the best way is to crawl the most important pages before the least important ones. For example, focused crawler that concentrates on a particular subject, the web pages that have a high measure of similarity between them and the subject matter, will be crawled before pages that are less similar or not identical to the subject. This case is called "importance metric" in web crawler, it is a measure used to prioritise URLs in the frontier to help the crawler crawling impor-

tant pages early (Cho et al., 1998). Thus, if we have a web page p , one of the following metrics can identify its importance $I(p)$:

- Backlink Count $IB(p)$: the number of pages pointing to the page p .
- PageRank $IR(p)$: a value or weight assigns to all links over the web where the importance value of web page p is equal to the total number of importance value of web pages pointing to it.
- Forward Link Count $IF(p)$: number of outgoing links from the page p .
- The Similarity to a Specific Topic $IS(p)$: the value of measurement of similarity between web page p content and the topic keywords provided by a user.

based upon, the higher the number of back or forward links or the similarity of the page, the more important the page becomes. However, in this study, we will use web page dynamicity combined with its similarity value as an importance metric to prioritise URLs in the frontier.

1.1 Problem Statement

As for the natural dynamicity of the web pages, the crawler needs to revisit them frequently to detect whether they have changed or not. When the crawler starts recrawling, three issues need to be considered:

1. Time Limitation: since the crawler needs to revisit previously crawled pages, therefore, after a certain period of crawling, the crawler has to be terminated to start recrawling the old pages again. Hence, at each crawling process, as the time is being restricted, some pages may be not be recrawled.
2. Network Problems: the network interruption that could be happened during the crawling process, resulting in crawling on certain number of pages.
3. Dynamicity of the Web Pages: There are two types of dynamic web pages. One of them changes very frequently like news web pages. The other one does not change much, and may take months or years to do so.

Consequently, the crawler must decide which are the highest dynamic pages to crawl, before the less dynamic ones.

1.2 Goal and Objectives:

The main goal of this study is to define an importance metric to measure how dynamic a web page is and use it as an ordering schema to help the crawler prioritise URLs inside the frontier from

the highest dynamicity to the lowest. As high dynamic web pages have a high probability of changing, this aim will be accomplished by performing the following objectives:

1. Designing a Focused Web Crawler with Two Importance Metrics:
 - a. Similarity Metric $IS(p)$: the Similarity measure between a web page and a specified topic or a set of topics. As news web pages are more dynamic than other pages, and since this dissertation focuses on web pages dynamicity, the designed crawler using cosine similarity measure will focus on specific categories of news pages.
 - b. Dynamicity Metric $ID(p)$: a numerator defines how many times the web page has been changed, through recrawling web pages recursively and detecting the changes that appear in them.
2. Prioritising URLs in the frontier from the highest dynamic to the lowest based on their combined metric $IC(p)$ of the similarity and dynamicity Metrics, as in the equation 1.1:

$$IC(p) = IS(p) + ID(p) \quad (1.1)$$

Thus, web pages that are categorised as news web pages and have a high dynamicity would be one of the most important pages and will have a high priority.

1.3 Scope of the Study:

This dissertation focuses on crawling web pages, and how to use their dynamicity score in order to prioritise their URLs in the frontier.

1.4 Dissertation Outline:

After clarifying and discussing the crawler with an overview, the following points illustrate an outline of the remaining content of this dissertation:

Chapter 2: discusses a set of publications related to topics covered in this dissertation, including focused web crawler, web page change detection policies and the priority of URLs in frontier.

Chapter 3: describes the methodology of the proposed work for prioritising URLs in the frontier based on their web pages dynamicity in order of crawling high ranked web pages in a very short duration.

Chapter 4: explains the details of the implementation of the proposed method explained in the third chapter.

Chapter 5: presents the results of implementing the proposed method.

Chapter 6: presents the conclusion and future work.

1.5 Summary

In this chapter, a brief description of web crawlers and their problems has been introduced, and the research problem is presented, mainly, how the crawler should revisit the changed pages before the constant ones. In favor of solving this problem, a method to crawl web pages based on their dynamicity has been proposed.

Chapter Two

Litrature Review

This chapter summarise the applications attributed to the topics that are included in the dissertation. The chapter is therefore arranged as follows: section 2.1 covers the architecture of the general web crawler, section 2.2 reviews the architecture of focused web crawler, section 2.3 presents algorithms that are used to discover the changes that appear in web pages, section 2.4 summarises the publications about different algorithms used to prioritise URLs in the frontier, and lastly section 2.5 concludes the whole chapter.

2.1 General Web Crawler

Web crawler is a computer program, which explores the World Wide Web to collect web pages in order to be stored, summarised and indexed by search engines and later to serve users' queries. (Udapure, Kale and Dharmik, 2014) provides a description for the general structure of the web crawler and the different strategies followed by the crawlers. Thus, Crawlers have three main components, which are:

- ♦ **Frontier:** a queue used to collect URLs to be crawled later based on a specific ordering schema. The frontier is initiated by a set of seed URLs that are provided by a user. Web pages of these URLs are downloaded and parsed. Also, their URLs are extracted in order to be added to the frontier. Moreover, the process of adding URLs to the frontier continues up to a specific number of URLs or until the crawling process is being interrupted due to the time or other network issues.
- ♦ **Page Downloader:** fetching web pages from the web, through sending an HTTP request with a URL to a web server and receiving a response within the web page. The time between the request and response should be restricted to ensure that the crawler does not consume its time waiting for the page, in case of weak connections.
- ♦ **Web Repository:** is a place where web pages are stored and accessed after they have been crawled. Likewise, all web pages are stored as HTML pages with a unique identifier in the repository.

Figure 2.1 illustrates the outline architecture for the working scenario of the crawler.

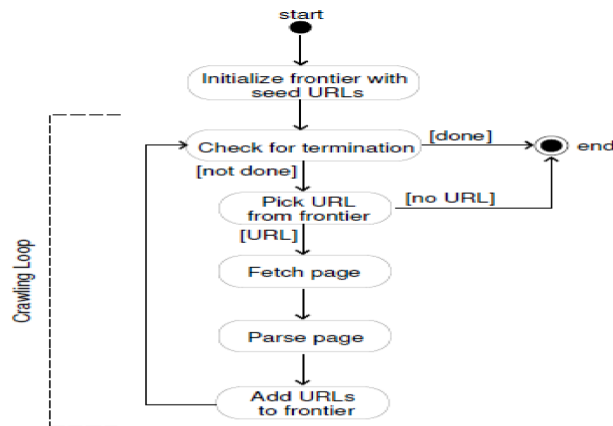


Figure 2.1: General Architecture of Web Crawler (Udapure et al., 2014)

Some examples of web crawlers are:

- Yahoo! Slurp: web crawler for yahoo.
- Bingbot: Microsoft's Bing web crawler.
- FAST Crawler & PolyBot: they are distributed crawlers.
- RBSE: the first published web crawler.
- Googlebot: Google's web crawler.

In addition, (Udapure, Kale and Dharmik, 2014) stated the different strategies that are followed by the web crawlers based on the way they are crawling the web pages, are as follows:

- ♦ Focused Web Crawler: web crawler that is assigned to crawl and collect web pages relevant to a specific topic and to neglect other pages.
- ♦ Incremental Crawler: instead of crawling each time from the beginning, the Incremental Crawler revisits the previously downloaded web pages recursively, in order to exchange old web pages with updated web pages that are presented on the web.
- ♦ Distributed Crawler: a set of crawlers working together, to perform the crawling process on web pages, in order to get from the web as many web pages as possible.
- ♦ Parallel Crawler: a web crawler, that leads a group of crawling processes at once, will result in a reduced crawl time.

(Udapure, Kale and Dharmik, 2014) conclude from the comparison they performed on different crawling strategies that, focused web crawler is designed to a particular set of web users, as it crawls a set of relevant web pages that are related to a specific topic and it saves the crawler's time and network resources.

2.2 Focused Web Crawler

The web contains a huge number of web pages and still, this number is increasing. For this reason, focused web crawlers that serve a particular set of developed users, are downloading only web pages relevant to a specific topic or set of topics, while discarding the others.

(Batsakis et al., 2009) mentions that focused web crawler is acquainted to meet the requirements of particular users for example, an expert domain or organisations through guiding the crawler to collect specific web pages. (Batsakis et al., 2009) states three approaches for building focused web crawlers:

- ♦ **Classic Focused Crawlers:** is a web crawler that takes a user's query as an input, and uses a model of information similarity as in: boolean model or the Vector Space Model (VSM) in order to compute the similarity between the user's query and the web page text. Thus, the greater the value of the similarity, the higher its priority to crawl. Typically, focused crawler URLs are prioritising in the frontier based on their similarity value between the user's query and either between their anchor texts, or the parent web page of the URL.
- ♦ **Semantic Crawlers:** semantic crawler is another kind of classical crawlers, where web pages are relevant to the user's query in case they are conceptually related. In addition, ontology is used to acquire all terms that are conceptually similar to the topic terms. As in classic crawlers, semantic crawlers' similarity is calculated by the Vector Space Model or by specialized models such as the Semantic Similarity Retrieval Model (SSRM) and URLs prioritising in the frontier depending on the semantic similarity either of their anchor text or the parent web page.
- ♦ **Learning Crawlers:** are web crawlers that apply a classification process of web pages through providing the crawler with a set of relevant and non-relevant web pages, in order to train it to classify and assign priorities to web pages. In addition, learning Crawler does not only classify web pages according to the content of them. However, the structure link that leads to relevant

pages is considered as well. Furthermore, many methods are used in Learning Crawler such as Context Graphs and Hidden Markov Models (HMM). In (Batsakis et al., 2009) paper, a new HMM model is proposed where pages were classified as relevant, by computing the similarity between its text and the most relevant (positive) pages to the specified topic in the provided training set.

(Batsakis et al., 2009) implement these three crawlers and evaluate their performances using Harvest rate, in which it evaluate the crawlers' performances based on the number of the relevant web pages that are downloaded by the web crawler. Additionally, they compare the crawlers' performances in terms of their ability to download relevant web pages by prioritising URLs in the frontier based on three cases:

- Prioritising based on page content.
- Prioritising based on URL anchor text.
- Prioritising based on combining page content and URL anchor text.

The results of the three crawlers demonstrate that using anchor text outperform the page content, as the anchor text summarises the web page more than its content. However, the crawlers show their best performances when they merge the page content and anchor text. Furthermore, (Batsakis et al., 2009) make a set of comparisons between the crawlers, in order to analyse their performances, and the comparison is performed among:

- Classical crawlers and semantic crawlers.
- HMM crawler and the proposed HMM crawler.
- HMM crawler and classical crawlers.

In all the three comparisons, URLs are prioritised by combining both pages and anchor text. In addition, the results illustrate that the classical crawler outperforms the semantic crawler and the proposed HMM crawler outperforms HMM crawler. Nevertheless, the performance of the last comparison is differentiated according to the ambiguity of the topic terms, which means that in case of a topic that is consisted of clear terms, classical crawler outperforms the HMM crawler. Yet, in case of topics with ambiguity terms, HMM crawler outperforms the classical crawler.

(Bhatt, Vyas and Pandya, 2015) present a study about the focused crawler architecture and the different algorithms used to increase the focused crawler efficiency. (Bhatt, Vyas and Pandya,

2015) define the focused web crawler as a hypertext system that crawls and preserves web pages, which are related to a specific topic or a set of topics. This process leads to saving network resources like CPU, network bandwidth, as well as reducing the crawling time.

The architecture of the focused web crawler is similar to the general crawler structure. The only difference is the classifier component, which is used to decide the relevance of the web pages. Moreover, the algorithms which have the ability to be implemented by the focused crawler in order to increase the accuracy for estimating the web page relevance are:

- **Best-First Search:** in this method, each time the crawler needs to extract a URL in order to be crawled; it examines the entire frontier looking for the best URL that needs crawling. By using a heuristic function, a priority value is assigned for each URL in the frontier where a URL with a high value will be given a higher priority to be crawled next.
- **Fish-Search:** the crawler in this algorithm extracts a URL from the frontier that crawls its web page and parses its content to be handled by a binary classifier, which uses zero and one values. This is to indicate whether web page is relevant or irrelevant for a given topic.
- **Shark-Search:** an extended version of fish search. However, Shark-Search estimates page relevance with a real number between zero and one.
- **Genetic Algorithm:** the major goal of this algorithm is to optimise the quality of the retrieved relevant web pages by the focused crawler. This algorithm uses the Jaccard similarity function, in order to measure the relevancy of web pages between the searched results that are obtained by the focused crawler.

Moreover, (Bhatt, Vyas and Pandya, 2015) explain in their study some challenges to deal with while using the focused web crawler, as in missing relevant pages, keeping the database updated through refreshing its web pages and finally the absence of context in focused crawler, which results in increasing the number of irrelevant pages.

2.3 Detecting Changes in Web Pages

There are four types of changes that usually appear in web pages: structural, textual, presentation and behavioral changes. However, the proposed method will limit its focus on structural and

textual changes.

For textual changes, (Singhal and Sharma, 2012) presents a new method to discover and detect efficiently the changes that appear in the web pages' text. However, their proposed method focuses only on the textual changes and pays no attention to whether the page has structurally changed or not. In addition, detecting the changes between the old and new pages are performed in two steps. Firstly, by using a Unicode system in which a code is assigned to all words that appear in both pages. Secondly, by performing a word-by-word comparison process where each word in the old page is compared with the corresponding word in the new page, and whenever there is a difference between two words, the comparison process is terminated and the old page is replaced with the new one. Otherwise, the comparison is continued, until the end of the web page.

The method proposed by (Singhal and Sharma, 2012) illustrates the following benefits: first, it works with all natural languages and all characters since it is using the Unicode system. Second, it saves time and memory, as it ignores the structural changes. In addition, the comparison process stops whenever it detects any change.

Furthermore, as for detecting the structural and textual changes, (Goel and Aggarwal, 2012) propose a new algorithm for detecting the structural, as well as textual changes of the web pages. This proposed algorithm is divided into two parts: a tree development part and a change detection part. In the first part, web pages have to be fetched, both the new web page from the web and the old web page from the repository. In addition, the tree has to be developed for both pages accordingly, through extracting the tags presented in their HTML code. Whereas, in the second part, the comparison process that is concerned with detecting the changes will be performed through assigning values for all the tags extracted from HTML code, which are considered as nodes in the developed tree.

Nevertheless, in the case of leaf nodes, the algorithm assigns hash value by using the tag number and level number where the node is located in the tree. However, the tag value of the non-leaf nodes is equal to the sum of hash values of their sub nodes. As a result, all nodes of both trees are being compared. Moreover, if there is any difference noticed in the value of any one of the tags, this means there is a change occurred in the pages and the repository needs to be updated.

The study concludes that, this algorithm is beneficial for saving the time and providing the

user with the changes that occur in a web page. However, the drawback of this algorithm is that, any changes that may occur in the node text will not be noticed, due to the fact that the algorithm is only operating in case an adding or deleting happens in the node itself and not in its content.

2.4 Prioritising URLs in the Frontier

Revisiting web pages in order to detect changes, it is a necessary procedure performed by the crawler. Also, due to the problems faced by the crawler, the crawling process is terminated while some pages have not been crawled yet. Since web pages vary in their importance for a crawler, many algorithms have been developed to define which pages are the most important to crawl first according to the crawlers, and that is by giving their URLs a priority in the frontier.

(Cho et al., 1998) make a comparison between different importance metrics used by the crawler to make a decision which web page is the most important to be crawled earlier than the other web pages. In their study, the crawling process performs over 179.000 pages in a single domain (Stanford.edu) website, and these set of pages has been crawled and stored earlier by the authors. (Cho et al., 1998) compare three importance metrics: Breadth-first, Backlink-count, PageRank. In addition, they evaluate the metrics based on how fast they are in leading the crawler to hot (important) pages. The results of the study show that PageRank has the ability to crawl hot pages earlier than others do.

Furthermore, (Cho et al., 1998) conduct two experiments about the ability of these three metrics to crawl earlier the most relevant or hot pages assigned to a specific topic. The first experiment is performed by prioritising URLs based on the relevancy of their anchor text and the second experiment is based on the anchor text, the URL similarity or whether the web page is within three links from a hot page. The results indicate that the performance of crawlers has significantly increased in the second experiment by using the anchor text, URL or the link distance between the web page and the hot web pages.

(Choudhary and Roy, 2013) suggest a new crawling algorithm about prioritising URLs in a focused web crawler, through assigning values to URLs using semantic similarity. Thus ,the suggested algorithm is performed as follows:

1. Seed URL is fetched from the frontier.
2. Web page corresponding to the URL is downloaded.

3. URLs extracted from the web page and their corresponding web pages are downloaded.
4. Calculating the semantic similarity using cosine similarity, where the semantic score between every downloaded web page and a specific topic ontology are calculated by combining three similarities which are the URL anchor text, the web page text, and the parent web page.
5. Extracted web page URLs and their semantic scores added to the frontier.
6. Crawler extracts the URL that has the maximum semantic score from the frontier.
7. Steps 2_6 are repeated until reaching a specific number of URLs in the frontier or until it becomes free of URLs.

In order to measure how efficient the proposed semantic crawler when crawling relevant pages and filtering irrelevant pages, Harvest Rate is used, to evaluate and compare the performance of the proposed crawler with a simple, focused and priority based focused crawler. The results show that the crawler gives 88% more improved results over the simple crawler, 28% more improved results over the focused crawler and 6% over priority based focused crawler.

(Kumar and Jain, 2014) propose a new method, which is a query-based approach using sitemap file to alert the crawler about the modified web pages with the highest priority. Additionally, in the proposed method, the crawler sends HTTP request to the website with the LAST_VISIT parameter that is indicated, when the crawler perform the crawling on web pages in the last time. After that, the crawler receives a response with a list of the highest priority URLs that are updated after the crawler's last visit, in order to be downloaded. Furthermore, experiments show that this method only downloads the updated web pages, which leads to a reduction in using network resources and traffic.

(Baker and Akcayol, 2017) present a new algorithm to prioritise URLs inside the frontier, for guessing the importance of web pages based on dividing webpages' URLs into inter and intra links. In their suggested method, extracted URLs are separated based on their domain into two categories after fetching and parsing a web page. Intra links which are links that refer to web pages of the same domain as their parent web page domain and inter links are differentiated on the parent web page domain. Thus, the algorithm assigns weights to both categories, given $2/3$ for inter links, $1/3$ for intra links. The reason for giving a higher weight for inter links is to avoid link-loops inside

the domain. However, inter and intra links and their weights are used to estimate the importance of the web page (link score (LS)) in a calculation based on the equation 2.1:

$$LS = \frac{\Sigma(\alpha) - \beta}{\Sigma(\alpha)} * \theta \quad (2.1)$$

Where α is the total of inter and intra links, β is the least number of links between inter and intra links and θ is a selectable value representing either the weight of inter or intra links. After the web page is parsed and the link score (LS) value is calculated, all of its extracted URLs are added to the frontier. All extracted URLs are added to the frontier according to their parent web page's LS value in descending order. Therefore, the selection of the next URL that is supposed to be extracted from the frontier will be the URL that has the highest score. In addition, their proposed algorithm uses a timing mechanism to avoid URLs from being in a state of waiting inside the frontier for an extended period. Thus, this mechanism drops any URL from frontier in case it exceeds the maximum waiting time inside the queue. The experimental results show that the developed algorithm gives a well-crawled performance.

2.5 Summary

To conclude, the focused web crawler is implemented as a general crawler except that they have the classifier component to decide whether the web pages are relevant to the specified topic or not. In addition, some of the classifiers use Vector Space Model, such as classical and semantic crawlers, while others use a training set to guide the crawler during the classification process in the learning crawlers. Moreover, different algorithms, which are used to detect changes appear in web pages, some are detecting only textual changes, while others are working on both textual and structural changes. Finally, many algorithms are presented for prioritising URLs in the frontier in order to crawl earlier the most important pages, some of which depend on similarities between web pages and a specified topic as in focused web crawlers, others depend on the incoming or outgoing links from the web page and other algorithms depend on the sitemap file. However, all of them show impressive results when crawling earlier the most important web pages.

From all the methods illustrated in this chapter, the following methods are going to be used in this dissertation:

1. Creating classic focused crawler using the Vector Space Model.
2. Computing the similarity of web pages through calculating the average of the two similarities the web page text and its anchor text.
3. Developing trees for both old and new web pages in order to detect changes that happened to them.

Chapter Three

Methodology

We introduced, in this chapter the proposed method of using web page similarity combined together with its dynamicity as an importance metric in prioritising URLs in the frontier.

The proposed architecture in this dissertation consisted of three layers that focused more on detecting changes of web pages and ordering URLs in the frontier:

- A. As for the first layer, URLs were provided and inserted into the frontier in descending order. However, a method of prioritising URLs in the frontier is based on the combined metric of similarity score and dynamicity of web pages was considered.
- B. The second layer, which supported the decision of web page relevance, where web pages were crawled and classified into one of the news categories, and they were: Business, Politics, Crime, Health, Sports, Technology (Bracewell, Yan, Ren and Kuroiwa, 2009). The classification of web pages was made using the Vector Space Model (cosine similarity measure) since is widely used in the information retrieval system, which was discussed by (Lee, Chuang, and Seamons, 1997).
- C. Within the third layer, the web pages were crawled for a period of time and changes were detected to be used later by the first layer in defining the dynamicity of web pages, and in order to crawl most dynamic web pages first. Furthermore, details on these layers were shown in the following sections.

The interconnection and working process between these layers were shown in Figure 3.1.

As for the proposed research work, it went through the following phases:

1. First: Creating focused web crawler and classifying web pages.
2. Second: Recrawling and detecting changes in web pages recursively.
3. Third: Prioritising URLs in the frontier.
4. The final phase was the evaluation phase, which depended on checking what was the priority of the web pages in the other methods used by the crawler in prioritising URLs in the frontier.

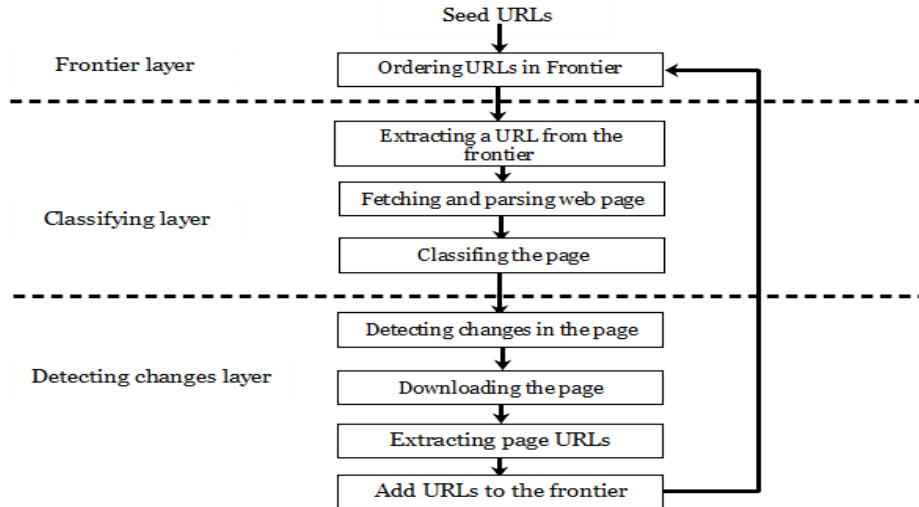


Figure 3.1: Overall Crawling Process

3.1 Focused web crawler

As stated earlier in chapter 2, the focused crawler is a general crawler with a classifier component (Bhatt et al., 2015). Hence, the process of creating a focused crawler composed of two main processes, firstly the crawler, which fetching and crawling web pages. Secondly, the classifier, which classifying web pages into their relevant classification. However, a focused web crawler required a set of terms to represent the classification, a threshold value to decide whether a web page relevant or not, fetching and handling the web page content, a process of classifying the web page, reading the robot file, extracting the web page URLs and finally add the page to the repository. Further illustration of focused web crawler is presented in sections 3.1.1 to 3.1.7.

In this dissertation, some crawler issues, such as DNS caching and check if URL has content already seen are disregarded as they are not necessary for the proposed crawler.

3.1.1 Terms Selection

To classify web pages, two sets of terms represented as vectors were required. The first set was a group of vectors that represented news categories in each, whereas the other set was a vector that represented web page documents (discussed later in section 3.1.5). There were two ways to prepare a vector of terms that represented any category at all (Assis, Laender and Gonçalves, 2008):

1. Manually: It involved the assistance of experts whose were familiar with the topics of classifications.

2. Automatically: It performed by applying a semi-automatic generation method; this strategy could be applied if we owned a set of URLs, whose pages described the classification of interest. Semi-automatic generation method was implemented in this dissertation to generate vectors of news categories. In order to generate these vectors, web pages, which were expressing news categories, were required to be collected, and we used two methods, the first method proposed by (Assis et al., 2008) and the second method is proposed in this dissertation:

1. First Method: implemented using a search engine. It was made only once before the start of crawling web pages. In this method each category was issued as a query in the search engine. And a set of URLs that represented the category were selected from the retrieved results.

2. Second Method: implemented using irrelevant web pages. It was made throughout the crawling process for each news category. Typically; there were relevant and irrelevant web pages. Relevant web pages were accepted, while irrelevant web pages were dismissed. Within this dissertation, irrelevant pages were classified into two categories:

- a. Irrelevant pages out of the classification, which were neglected.
- b. Irrelevant pages belonged to the category, but their similarity measure was less than the defined threshold (discussed next in section 3.1.2). This kind of pages wasn't neglected; instead, they were assessed and handled to extract the most frequent words.

After collecting URLs, whether from the search engine or throughout the crawling process, the process of choosing common frequent terms was implemented as following (Assis et al., 2008):

- 1. Term Table: a table created to store the number of occurrence of all terms founded in the web page text.
- 2. Fetch and Parse Page: for each collected URL its web page were fetched and parsed, in order to create a list (L) contained web pages' terms and their frequency.
- 3. Update term Table: including the terms in the list (L) into the term table in case they had not inserted before or updated their frequency in case they had been previously added.
- 4. Filter Terms (N): this step was responsible for selecting the (N) most frequent terms in the term table.

5. Generate News Category Vector (q_i): added (N) terms returned by the previous step into news vector(q_i).
6. Steps (1)-(5) were implemented with every set of collected URLs that represented the news categories.

3.1.2 Determining Similarity Threshold Value

The similarity threshold value is defined as a measure used by focused web crawlers to determine either a web page was relevant to the specific topic or not. After determining similarity value of a web page to a news category, this value was compared by the specified threshold value and if it was greater than or equal to the threshold, web page was accepted and considered as relevant, otherwise it was neglected.

(Siqueira et al., 2017) had proposed many strategies to determine the similarity threshold. A strategy based on seed pages summarisation was the method that had been chosen to be implemented in this dissertation. This strategy worked by calculating the arithmetic mean of similarity values of a set of web pages expressing the classification of interest. However, these web pages were collected by two methods:

1. First Method: applied via search engines by submitting the topic of interest as a query in the search engine.
2. Second Method: implemented by using relevant web pages that had been visited by the crawler at the preceding crawling phases. In this method at each crawling phase, the previously crawled pages from all the previous phases with their URLs and similarity values were retrieved to calculate the arithmetic mean of their similarity values.

At the first crawling phase and since there were no pages crawled yet, the similarity threshold were determined using the first method, while at another crawling phases, the threshold value were performed with the second method.

3.1.3 Fetching Web Pages

Within this stage, a web pages were retrieved and tree modules of the web page were built and processed. Hence, the fetching process consisted of three processes:

1. Fetching Process: a process responsible for extracting URLs from the frontier and fetching their

pages. The Fetching process was implemented by the regular HTTP transaction, with which the client sent an HTTP request for a web page to the web server and the web server then would answer by the web page in return.

2. Tree Builder Process: following the fetching process, the web pages were sent into the tree modules builder, to create the Document Object Model (DOM) of the web page to represent the web page as a tree structure to be accessed and manipulated (Wood et al., 1998). In this process web pages were represented as a tree structure of tags. Through extracting HTML tags of their pages, where all elements represented with tags such as <body>, <table> or <p>, became nodes.
3. Removing Unnecessary Tags: cleaning up web page tree through removing scripts, style and comments.

3.1.4 Parsing Web Pages

At this process web page tree was processed in order to extract the text of their tags to be used in classification later. This process included the following procedures:

1. Removing HTML Tags, keeping only the text of web page, by extracting textual nodes which are leaf nodes representing as #text node in the DOM structure and its value is a text string.
2. Tokenizing Web Page Text: web page text was split into tokens based on white spaces, delimiters and added into a list (T).
3. Removing Stop Words: stop words are common words, such as prepositions and conjunctions (Capella University, 2007) that have low importance later in classification process. This process was performed by comparing every word in the list (T) with a previously prepared list of stop words. A List of common english stop words is shown in Table 2 in Appendix (A).
4. Word Stemming: stemming the words in the list (T). It is a process of gathering a set of morphologically similar words, such as *implement*, *implementation* and *implemented*, which were all normalised as *implement*. In this dissertation, the stemming process was performed by implementing Porter Stemming Algorithm, due to its accuracy and simplicity (Porter, 1980).

3.1.5 Classifying Web Pages

Subsequent to fetching and parsing, classification process had to be implemented; in this design, the web pages had to be assigned to one of the following news categories: Business, Politics, Crime, Health, Sports and Technology (Bracewell et al., 2009). As (Batsakis et al., 2009) mentioned

that, focused crawlers can be categorised as follows: Classic Focused Crawlers, Semantic Crawlers and Learning Crawlers. Within this dissertation, classical focused crawler design were brought about to classify web pages due to its simplicity and efficiency. Classical crawlers took terms vectors of news and web pages as input in order to assist crawlers to get to the pages of interest. The implementation of classical crawlers was performed as the following:

1. Web Page Representation: after extracting the text of web page, features that represented web page as a vector were selected. In this dissertation, Bag-of-terms representation method was utilised. Bag-of-terms is a method used to represent web pages as a vector of terms with their weights. Moreover, the basic concept of the method was that a web page was represented by a vector (d_i) contained the words (t_1, t_2, \dots, t_n) in the stemmed list T (figured through the section 3.1.4) together with their weights w_{ij} (or frequency), as shown in equation 3.1:

$$d_i = (w_{i1}, w_{i2}, w_{i3}, \dots, w_{iM}) \quad (3.1)$$

That is, M is the number of indexing words and w_{ij} ($1 \leq j \leq M$) is the frequency of word (t_j) in the web page d_i (Choi & Yao, 2005). There are two common methods for term weighting: term frequency (tf) and inverse document frequency (idf). The weight of a term in (tf) could be defined as the number of times a term happened in a web page while (idf): is the total number of crawled web pages divided by the number of pages where the term appeared. Hence, (idf) is the weight of term not in the page itself but also all the crawled web pages, which was considered a problem, where at each crawling process the crawler needed to refigure the weights of all the terms. In addition, it is worth mentioning that most classical web crawler implementations use term frequency (tf) weights to compute web pages relevance (Batsakis et al., 2009). Hence, each word in the web page vector (d_i) was presented by its term frequency.

2. Web Page Similarity: This procedure was concerned with computing the similarity measure between the web page vector (d_i) and the news vectors (q_1, q_2, \dots, q_n). The similarity between web pages and news vectors was calculated according to Vector Space Model by computing the cosine of the angle between web page vector and each one of news vectors which called cosine similarity measure. Cosine similarity calculated according to the equation 3.2 (Batsakis et al., 2009):

$$Similarity_{page\ content}(p) = \frac{\sum q_i d_i}{\sqrt{\sum q_i^2 \sum d_i^2}} \quad (3.2)$$

In this measure, q_i and d_i are the (*tf*) weights of the terms in news category vector and web page vector respectively.

3. Assigning the Web Page: a process of assigning web page to one of the news categories that possessed the highest cosine similarity score calculated from the preceding step.
4. Anchor Text Similarity: as in classic focused crawlers URLs prioritised based on their anchor text similarity value in the frontier, where the similarity between the web page anchor text and news categories vectors calculated. Anchor text similarity value was used also in calculating the total similarity of web pages as shown in the next step.
5. Combining the Page and Anchor Text Similarity Value: calculating the average of cosine similarity value between the web page p and its URL i anchor text similarity values as in equation 3.3 (Batsakis et al., 2009):

$$Similarity(p_i) = \frac{1}{2}(Similarity_{page\ content}(p) + Similarity_{anchor\ text}(i)) \quad (3.3)$$

After fetching, parsing and computing web page relevance, these operations were followed by subsequent operations which were reading Robot file, extracting and attaching URLs of the web page to the frontier.

3.1.6 Reading Robot File

Robot file is defined as a text file created by a website administrator to manage the crawler visits to website pages. This file consisted of a set of commands (directives) to guide web crawler in crawling website pages, each command was placed in a line separately. Therefore, the commands that can be included in the robot file were:

1. User-Agent: the name of the search engine to which crawl rules should apply. Example: User_agent: Googlebot, indicated that the following commands had to be implementd on the crawler named Googlebot. User-agent command supported the use of the asterisk (*) meant

that the following commands applied to all crawlers and not to a specific crawler.

2. Disallow Command: a web page or URL that was not permitted to be crawled by the crawler. For each crawling web page there was a list called blocked list, which consisted of set of disallowed URLs that were created to be used later in extracting URLs of the web page to determine which URLs will be extracted and added to the frontier and which ones will be neglected.
3. Allow Command: a command indicated to web pages that are allowed to be crawled by the crawler.
4. Sitemap Command: represented a place where a sitemap file of the website located.

The following example illustrated a sample of these command in the robot file:

User-agent: Googlebot

Disallow: /URL1/

Allow: /URL2/

Sitemap: <https://example.com/sitemap.xml>

3.1.7 Extracting URLs of Web Pages

After determining the relevance of the web page and preparing the blocked URLs list, URLs extraction process was performed. In this process, URLs and their anchor text were extracted of the web page text string, through searching for anchor tags (`anchor text `) to extract their href attribute values and anchor text. After extracting process URLs normalised and attached to the frontier. Following procedures had to be performed before adding URLs to the frontier:

1. Canonicalisation URLs: a process of normalising the extracted URLs, involved transforming URLs into a canonical form, through transforming all relative URLs into their absolute form where:
 - a. Absolute URLs: URLs that involved the full path of the page included schema, domain name and the path. This kind of URLs would be added directly to the frontier without any handling.

- b. Relative URLs: these were URLs that did not include the full path of the web page, did not specify schema and domain name, only the path part of the URL referring that the URL and web page emerged from the same site. Relative URLs needed to be handled before adding them to the frontier.
2. Parsing URLs Anchor Text: as in web page text, anchor text was fetched, parsed and classified by computing their cosine similarity between their text and news terms vectors in order to use their similarity values in:
 - a. Prioritising URLs inside the frontier (discussed later in section 3.3).
 - b. Computing the average similarity of web pages between their text and URLs anchor text (discussed in section 3.1.5).

Following finishing these procedures, URLs were attached to the frontier, where every extracted URL was added besides its similarity value. In addition, every time a URL was added the frontier reordered according to the URL similarity value in descending order. Hence, when the crawler selected the next URL from the frontier, it would be the maximum URL. More about, the robot file and URLs processing was described in chapter four.

3.1.8 Updating Web Pages Repository

It was the last stage of crawling a web page. A web page was downloaded and stored in the repository and the URL with the similarity value was inserted into its news category table in the database. Both URL and web pages were saved with the same unique identifier. The overall crawling process of the classic focused crawler was shown in Figure A.1 in Appendix (A).

3.2 Revisiting and Detecting Changes in Web Pages

The deposited web pages needed recrawling recursively to keep the repository updated with fresh copies of web pages. In this step, structural and textual changes that the web pages underwent were detected, so they could be used later by the crawler in assigning priorities of URLs in the frontier. Since the proposed method depended on the dynamicity of web pages. Therefore, each web page had a structural counter and textual counter to calculate how many times the page had been structurally changed and how many times it had changed textually, so it could be combined later

and used to know how dynamic the page was. This process of detecting changes will be explained in the upcoming steps, using a certain method:

1. For all news category; the pages that had been crawled in advance, their URLs were retrieved from the table into the frontier in descending order and extracted one by one. (More will be discussed later in section 3.3).
2. For each URL the new web page was crawled and the old web page was retrieved from the repository.
3. Trees were developed for both pages.
4. Both trees were cleaned up from unnecessary tags script, style, and comments. Keeping only HTML tags in both trees.
5. Re-classifying web page. Web pages category were verified to see if it had changed or not, applying the classical crawler procedures again to the page presented on the web. In case the page category was not changed, then changes were detected otherwise if the page category had changed, it had to determine whether it was relevant to the category that it had changed to or not, that was if its similarity value was greater than or equal to the category defined threshold value. The following steps had to be performed in case of the page being accepted in the new category:
 - a. The webpage's data (structural counter and textual counter) were shifted to its new category.
 - b. The webpage was eliminated from the old category and changes were detected until the crawler needed to start crawling its category URLs.

Following these procedures and in the case of the web page category was not changed, both trees were moved into the next stage, which was detecting structural or textual changes.

3.2.1 Detecting Changes of Web Pages

There were two kinds of changes that were considered in this dissertation, and they were the structural and textual changes.

3.2.1.1 Detecting Structural Changes

This method captured the changes that resulted when a new tag was added or deleted in HTML code of a web page. Anyway, to take the structural changes into consideration, the next steps will be followed:

1. Both trees, new and old ones, were aligned with each other.
2. Every node in the old tree had been matched and compared to its corresponding node in the new one.
3. In case the type of both nodes was not identical (e.g: in old node type <div> but in new one <section>), or a number of children of both nodes was not equal, the URL's structural counter increased by one, and the search process for changes stopped and the structural counter incremented. Figure 3.2 illustrated this step.
4. During the absence of any structural changes, searching for textual changes commenced.

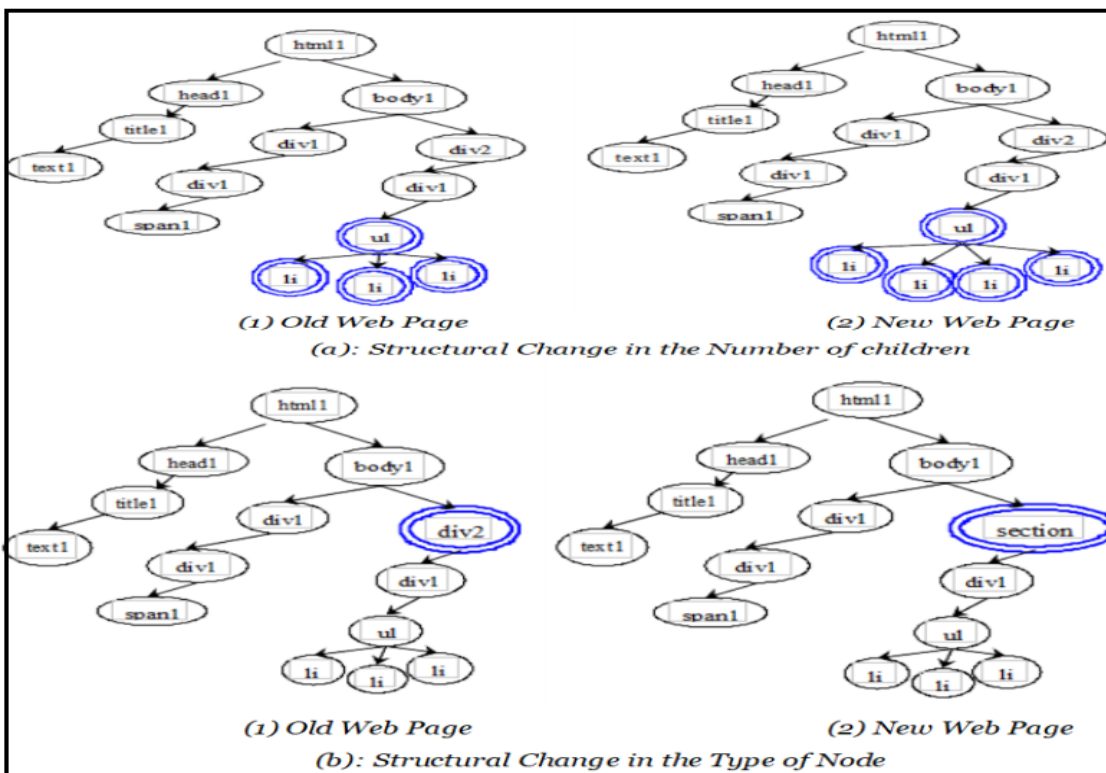


Figure 3.2: Types of Structural Changes in Web Pages

3.2.1.2 Detecting Textual Changes

This step was executed through the analysis of the text of HTML tags contained in both pages using the trees that represented them. In a certain case, textual changes were concentrated in particular sections of a web page, where these sections were believed to be more dynamic than the others which were considered static parts. Therefore, we proposed a new method that isolated these dynamic parts of the static ones and checked their texts separately. To accomplish this method, the crawler had to check whether the page had previously stored textual dynamic parts or not. As a result, to detect textual changes there were a number of cases and steps to take into account:

1. Case web page that had not previously stored textual parts:
 - a. Both trees, new and old ones, were aligned with each other.
 - b. All the textual nodes which were labeled as (#text) nodes in both trees were brought from both, old and new trees are aligned with each other since the page did not undergo through structural changes; so, both trees would had the same nodes.
 - c. The text of every old node was compared to the text of its corresponding new node.
 - d. If any node its text was changed, it should look back to check whether if any of its ancestor tag types was a structural tag (e.g: div, table, section), in which case its path (path used to arrive at nodes in the tree) was chosen; and this node marked as a dynamic textual part. However, this method of stored ancestors of the node, rather than the node itself, was valuable when there were many textual tags (e.g: span, hyperlink, and a paragraph), in which their first ancestor was (e.g: div). Therefore, rather than storing the path of each one of these three nodes, only one single path was stored together with its text, and if there was not, then the node's path was chosen. In both cases, path, its text, and a counter which created and initiated to one were added and saved.
 - e. Steps (c)-(d) were repeated until coming across all the nodes that were obtained during step (b) and saving their data in case their texts had changed.
2. Case web page had dynamic textual parts:

- All the paths that had been saved earlier and had the highest rate (counter) of changes were retrieved.
- The text of the old node was compared to its corresponding new node, using the path to reach its text.
- In case any of the node's texts had changed, their counter would increment by one. Otherwise, it was not incremented.
- Step (b)-(c), were repeated until all the paths that had been obtained during step (a) were encountered and their texts compared. See Figure 3.3, for further illustration.

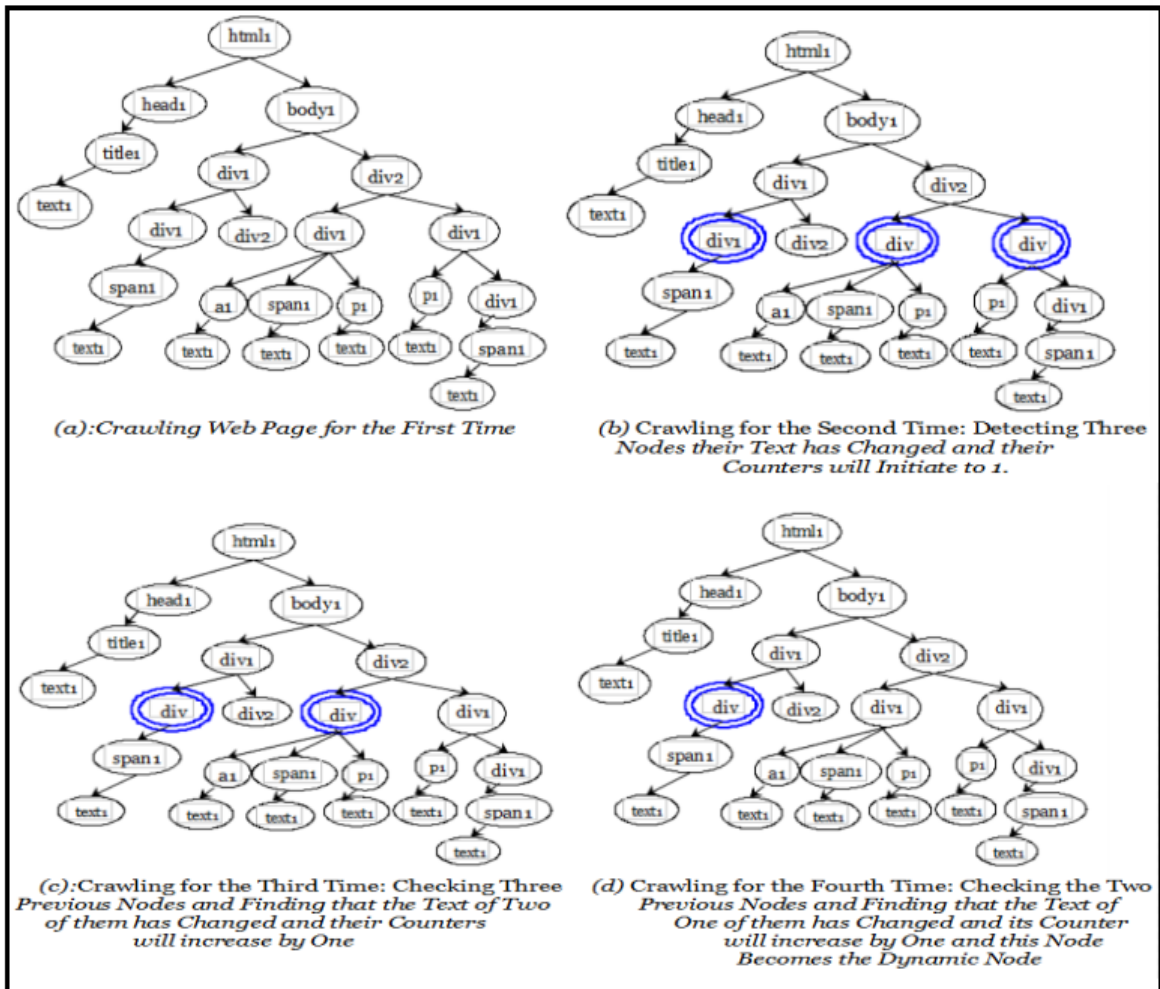


Figure 3.3: Show How to Create and Detect Changes in Textual Dynamic Parts of Web Pages

In case any one of the textual nodes had changed either in the first or second case, the web page textual counter was incremented. The method of dynamic parts could not be applied to structural changes. This was because when structural dynamic parts of web pages were checked whether they had changed or not, in case these parts did not change, while the other parts of the page did change; which was not examined by the way; because the scan was done just on structural dynamic parts. Hence, when the crawler started to detect changes in the textual dynamic parts, there was a problem, which was the path of these parts in new and old pages and which was not identical, therefore the textual comparison was not performed.

After the crawler completed recognising the changes of the web page whether they were structural or textual, the robot file read, URLs were extracted, and the old web page was replaced with the new web page in the repository. If the page had no changes then nothing happened and the crawler would move to the next page.

3.3 Priority of Web Pages in the Frontier (Ordering Schema)

(Adar, Teevan, Dumais and Elsas, 2009) discovered that the most popular and the most frequently viewed web pages by the users attended to be changed more than the less popular web pages. Based on that notion, we proposed to use web pages dynamicity as an importance metric in the crawling process, the more times the page changed, the more important it became. The proposed method in this dissertation used a combined metric $IC(p)$, using the similarity of web pages crawled by focused crawler together with their dynamicity, as in equation 3.4:

$$IC(p) = IS(p) + ID(p) \quad (3.4)$$

The similarity importance metric ($IS(p)$) was defined to be the textual similarity between web page vector and one of the news categories vectors. Web page dynamicity metric ($ID(p)$) was the total number of times the web page had changed structurally or textually. A web page that was relevant to one of the news categories and had a high dynamicity was considered one of the most important pages. Hence, every time a crawler was looking for the highest score URL, that URL carrying the highest $IC(p)$ was summoned. To implement this schema efficiently the following procedure had to be considered:

1. The crawler created two frontiers to visit: (1) Dynamic frontier: which kept URLs that had changes registered previously. (2) Static frontier: which kept URLs that had never undergone any changes, as well as the new URLs that had been drawn from either web pages that had changed or new web pages.
2. Ordering the Frontiers: dynamic frontier was ordered in descending order based on the dynamicity of its web pages just once at the beginning of the crawling phase. Static frontier was ordered based on the similarity of web pages; every time a URL was added to the static frontier, a reordering process was performed. The overall process of ordering URLs in both dynamic and static frontier and detecting changes in their web pages is illustrated in Figure B.1 in Appendix (B).

3.4: Evaluation Phase

On the contrary, other methods used by crawlers like *PageRank*, *Backlink Count* and *Forward Link Count* gave priority to the page based on its importance to the other pages. And even if the pages were static and unchanging, they were crawled first due to their importance to other pages. However, the proposed method in this dissertation, which examined the dynamicity of the page itself without its relation to other pages, gave the priority to changed web pages and postponed the static pages.

So *PageRank*, *Backlink Count*, and *Forward Link Count* were applied, to see what the crawl priority of pages which was considered the highest dynamic in the proposed method. The calculation of values from both methods *PageRank* and *backlink* was achieved at the level of all web pages' categories, that was because they indicated the total number of web pages referred to the web page.

At each crawling phase, web crawler recalculates the scores of the collected web pages and reorder them to be recrawled from the most important to the least. Therefore a time comparison between the *PageRank* and the proposed method in this dissertation were performed, to indicate how long both methods take to recalculate web pages' scores and reorder them at the beginning of each crawling process.

3.5 Summary

The prioritising and ordering of URLs in the frontier implemented in the proposed method by using a combined metrics which implemented by creating the focused web crawler and recognis-

ing changes appeared in web pages. Therefore, this metric combined between two metrics: (1) Similarity metric: created by implementing a classical focused crawler and assigning pages to their relevant category based on their similarity scores. (2) Web page dynamicity metric: determined by creating a counter for every page, the counter increased every time a crawler detected changes in the page. However, as for changes appeared in web pages only the structural and textual changes had been recognised. Detection of structural changes was performed based on the comparing of all tags in the web page HTML code, while textual changes performed based on the most textual dynamic tags of the web page.

At the first stage of each crawling phase, the crawler ordered URLs based on the proposed combined importance metric and split them into dynamic URLs and static URLs by creating two frontiers dynamic and static frontier, in order to give priority to dynamic web pages before the static ones.

Chapter Four

Implementation

In this chapter, we developed a web crawler that imitates the structure of a particular web crawler, which was described in chapter three by using C# as the main programming language as well as using SQL server in the storage features of web pages. In addition, experiments were implemented on a i7-core computer with 8GB of RAM running Window 10.

The remaining of this chapter is formed as follows: Section 4.1 exhibits the main data structures, and section 4.2 clarifies the main classes used in implementing web crawler architecture and finally, section 4.3 summarises the chapter.

4.1 Data structure

All the web pages' metadata and their websites were stored in the relational database tables. Likewise, the records of the tables accommodated all the information on web pages and sites except the web pages content, which were saved in folders based on their category, as every category had a folder representing it. Figure 4.1 shows the database schema. The Metadata of web pages for each news category is represented in two tables:

- ♦ Web page features table: the table includes the following data:
 - Web Page id: unique identifier signifying web page.
 - Web Page URL: URL of the web page.
 - Site id: unique identifier signifying web site.
 - Cosine Similarity Score: web page similarity value.
 - Structural Counter: representing the number of times a web page has structurally changed.
 - Textual Counter: representing the number of times a web page has textually changed.
- ♦ Web Page Dynamic Parts Table: the table includes the following data:
 - Web Page id: unique identifier signifying web page.
 - Part id: unique identifier signifying textual dynamic part of web page.
 - Part Path: the path of a textual dynamic part.

- Part Text: text of a textual dynamic part.
- Path Counter: number of time the path text has changed.

The Metadata of websites by which web pages can relate to, includes:

- Site id: unique identifier signifying website URL.
- Site URL: URL of the website.
- Number of URLs: number of web pages for each site.

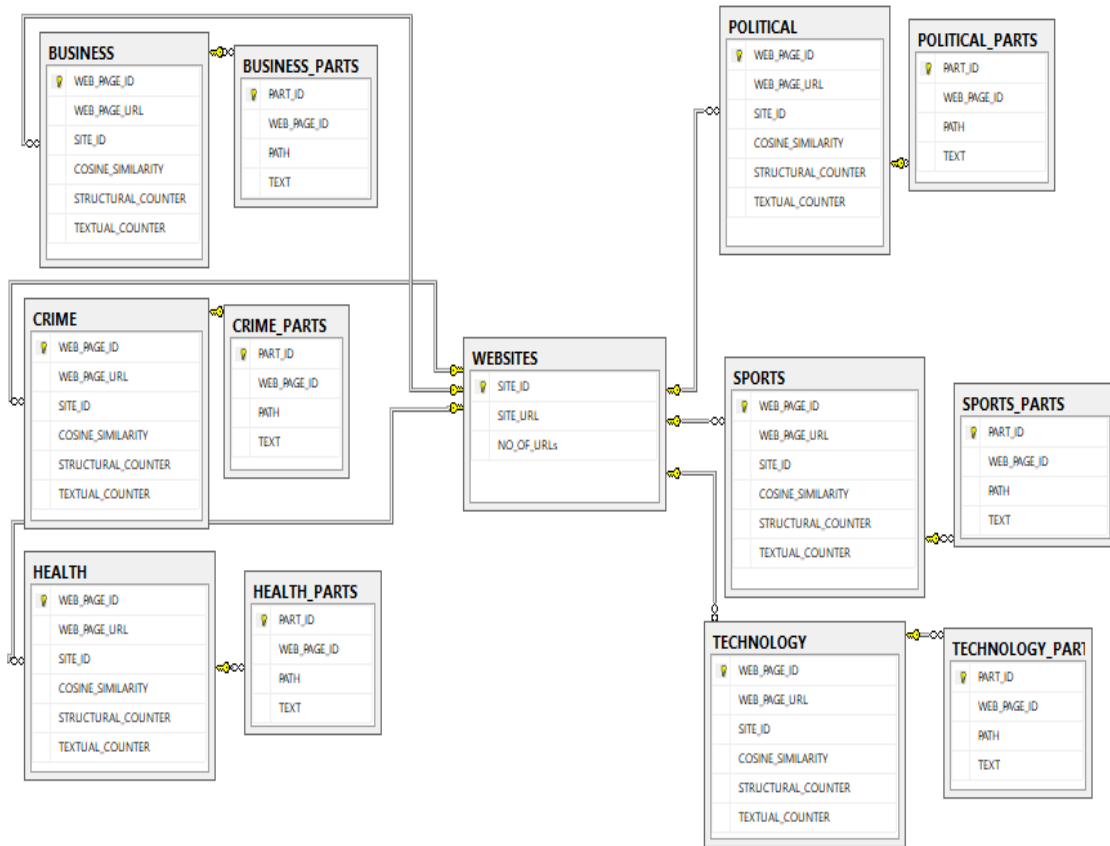


Figure 4.1: Database Schema

4.2 Classes

In this section, the main classes used in web crawler were represented in the following: fetching web page, parsing web page, terms selection, similarity threshold, cosine similarity, processing robot file, URLs extraction, detecting webpage changes and the frontier. Figure 4.2 illustrates the interaction between these classes.

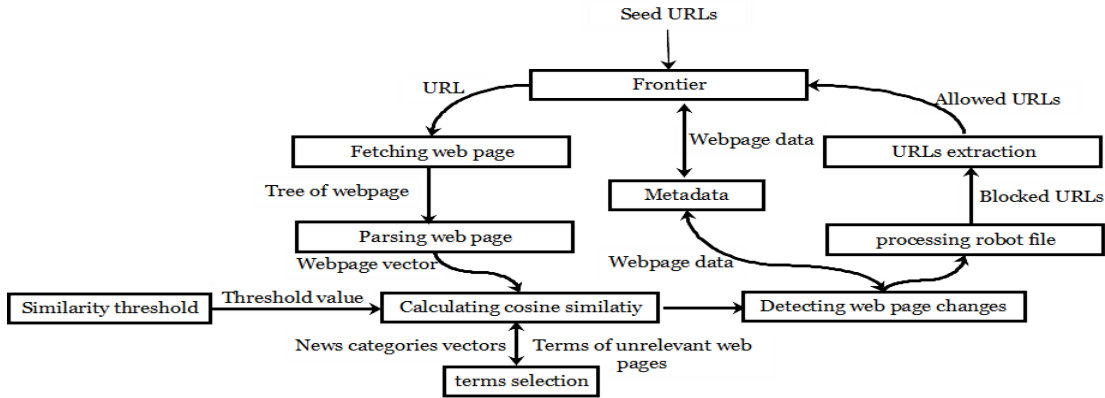


Figure 4.2: The Main Operation Steps of the Proposed Web Crawler

4.2.1 Class 1: Fetching Web Pages

Within this class, a web page would be retrieved and a tree module of the web page would be built, and it consisted of two modules:

1. Fetching Module: Extracting the URLs from the frontier in order to fetch their web pages from the web. The Fetching process of web pages would be implemented by the regular HTTP transaction.

Microsoft .NET framework possesses System.Net namespace, including `HttpWebRequest` and `HttpWebResponse` classes used in accessing internet resources. Additionally, the `WebRequest` object that transfers a URL identified the required web page through creating an HTTP request to the web server, and `HttpWebResponse` object had the web page as a data stream, which would be read and transformed into a string by the `Stream` and `StringBuilder` classes.

Furthermore, `HttpWebRequest` class had a timeout feature which was responsible for the number of seconds the request of the webpage should wait before the time was out, and in this dissertation, it remained for 60000 milliseconds (one minute).

2. Tree Builder Module: Following the downloading process, the webpage would be sent into the tree module builder, therefore:
 - a. Webpage text string would be represented as a tree structure of tags, where all elements were represented with tags such as `<body>`, `<table>` or `<p>`, became nodes.
 - b. Scripts, style, HTML tags and comments would be excluded from the tree. In some cases, latter HTML tags would be preserved.

Results of this class were a tree of the web pages with eliminating unnecessary tags.

4.2.2 Class 2: Parsing Web Pages

Parsing of the web page involved extracting words and URLs from the web page tree (URL extraction will be explained in section 4.2.7). In order to extract terms, following steps had to be implemented, and the web page (<http://www.bbc.com/sport/tennis>) will be used as an example to demonstrate the process:

1. Web page tree would be returned from class 1, and this process included eliminating all tags, which indicated that only text remained.
2. Web page text would be split into tokens based on white spaces and delimiters and would be added into an array (A). Figure C.1 in Appendix (C) shows a portion of tokens of the web page that amounts nearly 4856 tokens.
3. Eliminating stop words from array (A), where a list of common English stop words is shown in Table C.1 in Appendix (C).
4. Performing the stemming process on the remaining words in the array. Figure C.2 in Appendix (C) shows array (A) after eliminating the stop words and performing the stemming process, as we can notice the number of tokens (words) decreased to 351.
5. Creating a new array named as the web page vector, which saved each word and its frequency in the array (A). Figure C.3 in Appendix (C) illustrates the web page vector.

The output of this class revealed that web page vector contained each and every word and its frequency in the web page.

4.2.3 Class 3: Terms Selection

This class is executed to create a set of vectors that expressed news categories (Business, Crime, Health, Politics, Sports, and Technology). These vectors were created automatically without a specialist's assistance (as discussed in chapter 3 section 3.1.1). In order to generate these vectors, web pages, which were expressing news categories, were required to be collected, and two methods to collect the web pages were used.

The first method was implemented using the Google search engine. This method was applied only once before the start of crawling web pages and it is as follows:

1. Each category was issued as a query in Google, as for the sports category, the query was like (most important sports news sites).
2. Top 15 sites were chosen for the displayed results.
3. The first pages of each site were selected as web pages representing the category.

Table D.1 in Appendix (D) shows the selected pages.

The Second method, which was implemented using irrelevant web pages. This method were made throughout the crawling phases. Figure 4.3 display instances of web pages, which were related to various categories. However, they were categorised as irrelevant, since the similarity value was being less than the stipulated threshold.

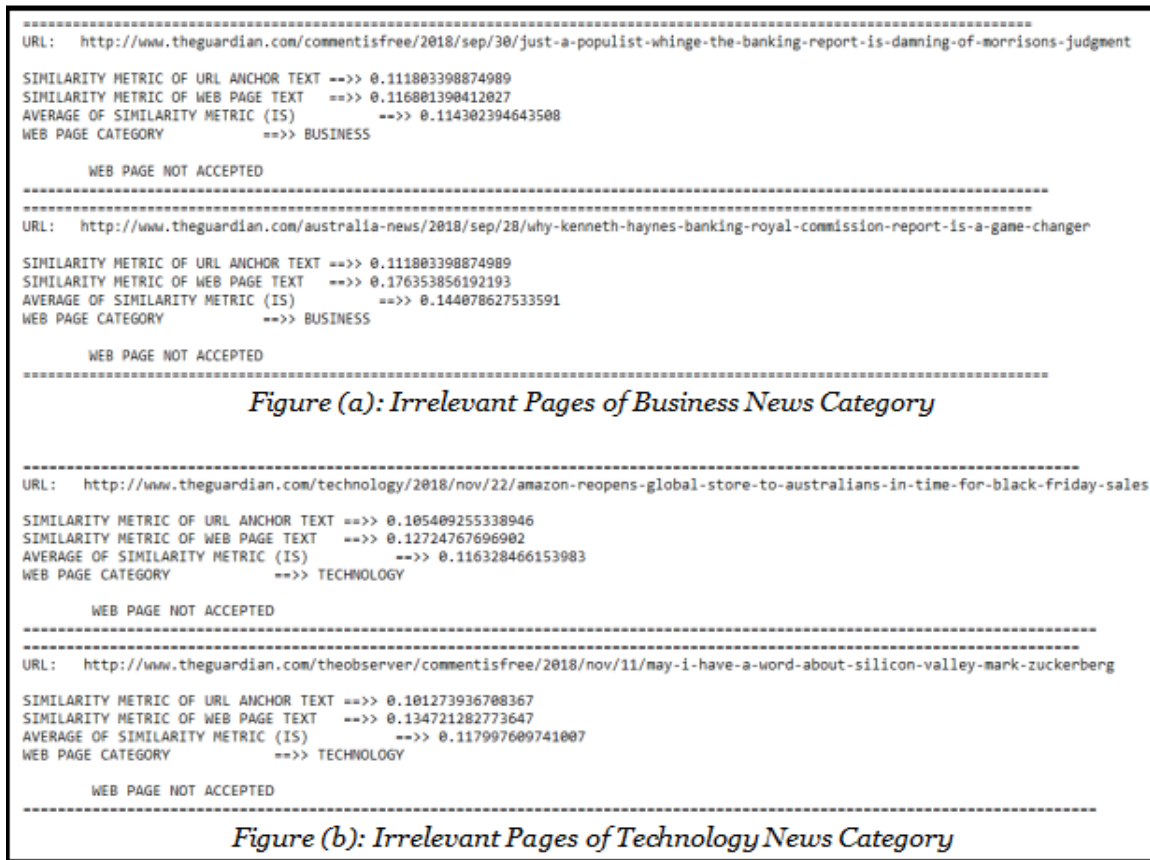


Figure 4.3: Neglected Relevant URLs

After collecting URLs whether of Google or throughout the crawling process, the process of chosen most frequent terms was implemented as following:

1. All of the collected URLs their web pages were downloaded and parsed by implementing class 1 and 2, and respectively returned with web page vectors holding stemmed terms of the web page,

and their frequency which was collected and saved in a table *L*.

2. The most frequent *N* terms were selected from the table *L*, where *N* differed from category to other because, after a particular number of terms, the importance of words decreased and did not express the category. For more about this process return back to chapter 3 section (3.1.1).

In addition, Table C.2 in Appendix (D) illustrates some of the URLs taken during the seventh crawling phase of crime news category. Furthermore, Figure 4.4 shows the top ten most frequent words which were derived from them. The idea behind choosing the top ten words was because of the words after the tenth term, the terms were considered normal and did not express the category; for instance: (Year, live, post, short, like, group... etc) they were not expressing the category.

```
-----  
WORD: pictur  ==>> FREQUENCY: 182  
WORD: investig ==>> FREQUENCY: 174  
WORD: stori   ==>> FREQUENCY: 161  
WORD: kill    ==>> FREQUENCY: 146  
WORD: report  ==>> FREQUENCY: 143  
WORD: head    ==>> FREQUENCY: 127  
WORD: asia    ==>> FREQUENCY: 126  
WORD: right   ==>> FREQUENCY: 125  
WORD: khashoggi ==>>FREQUENCY: 106  
WORD: europ   ==>> FREQUENCY: 146  
-----
```

Figure 4.4 : The Most Frequent Top Ten Terms from URLs in Table (2)

Result of this class was the terms that are representing news categories.

4.2.4 Class 4: Determining Similarity Threshold Value

As discussed in chapter 3 section 3.1.2 to determine the relevancy of web page to a news category, its similarity value had to be compared by the specified threshold value to decide whether the web page would be accepted and saved or dismissed.

In this class, the similarity threshold was defined to all news categories, by calculating the arithmetic means of their web pages. For the first crawling process, the similarity threshold was determined using the collection of pages displayed in Table D.1 in Appendix (D) that have been used in extracting terms, while in another crawling process, the calculating threshold was performed with pages that have been visited previously by the crawler. The following illustrations describe how to collect and prepare these pages to define the similarity threshold:

1. Case 1: Determining the threshold using collecting pages:
 - a. For each news category, URLs were gathered from Table D.1 in Appendix (D).

- b. Each URL was sent to class1 to download its web page and to class 2 to be returned by its web page vector.
 - c. Class 5 (classifying web pages) were implemented using the web page vector passed in step 2 to measure the similarity value of the web page.
 - d. Calculate the arithmetic mean of the similarity values of the web pages.
 - e. Steps (b)-(d) were repeated with each set of URLs that were representing the news category in Table D.1 in Appendix (D).
2. Case 2: Determining the threshold by using the previously visited web pages:
- a. URLs of web pages were retrieved from the database tables, each category individually.
 - b. Extracting the similarity values of all URLs and calculating their arithmetic mean. Figure 4.5 shows similarity thresholds of news categories used in the eighth crawling phase calculating by the similarity of the URLs previously crawled.

```

CALCULATING SIMILARITY THRESHOLD:
=====
BUSINESS SIMILARITY THRESHOD ==>>
NUMBER OF URL$ RETRIEVED OF BUSINESS TABLE ==>>    400 URL$
ARHMETIC MEAN (SIMILARITY THRESHOLD) ==>>    0.185463625
=====

CRIME SIMILARITY THRESHOD:
NUMBER OF URL$ RETRIEVED OF CRIME TABLE ==>>    135 URL$
ARHMETIC MEAN (SIMILARITY THRESHOLD) ==>>    0.168240592592593
=====

HEALTH SIMILARITY THRESHOD:
NUMBER OF URL$ RETRIEVED OF HEALTH TABLE ==>>    300 URL$
ARHMETIC MEAN (SIMILARITY THRESHOLD) ==>>    0.193716366666667
=====

POLITICS SIMILARITY THRESHOD:
NUMBER OF URL$ RETRIEVED OF POLITICS TABLE ==>>    478 URL$
ARHMETIC MEAN (SIMILARITY THRESHOLD) ==>>    0.19346859832636
=====

SPORT SIMILARITY THRESHOD:
NUMBER OF URL$ RETRIEVED OF SPORTS TABLE ==>>    581 URL$
ARHMETIC MEAN (SIMILARITY THRESHOLD) ==>>    0.191723993115319
=====

TECHNOLOGY SIMILARITY THRESHOD:
NUMBER OF URL$ RETRIEVED OF TECHNOLOGY TABLE ==>>    100 URL$
ARHMETIC MEAN (SIMILARITY THRESHOLD) ==>>    0.1718397
=====

```

Figure 4.5: Similarity Thresholds for Eighth Crawling Process

All in all, the results of this class displayed the threshold similarity values which was used by the crawler to decide whether to keep or to ignore the web page. Additionally, in case the page similarity was greater than or equal to the threshold, then the web page was saved and its URL was attached into its category news table, the robot file was scanned and the URLs were extracted and processed. Otherwise, the web page was neglected.

4.2.5 Class 5: Classifying Web Pages

Following downloading and parsing the page content, came the web page classification phase using the cosine similarity. The similarity of web pages was one of the metrics used in the proposed web crawler in prioritising URLs in the frontier. Cosine similarity was used to measure the cosine angle between two vectors, it is the dot product of two vectors that divided by the product of their lengths. However, in this dissertation, cosine similarity value was determined between web page vector and news vectors (created in class3). The Following steps show procedures what it would take to calculate cosine similarity and to categorise the web page that was continuing with (<http://www.bbc.com/sport/tennis>) web page (Manning, Raghavan and Schütze, 2008):

1. Web page vector was retrieved from class 2. Figure C.3 in Appendix (C) displays (<http://www.bbc.com/sport/tennis>) web page vector.
2. Calculating Scalar Dot Product value between web page vector X and news category vector Y. The dot product was a multiplication process of words frequency, which was made between every two similar words in both vectors and the words that did not have matching words were disregarded. The scalar product of vector X and vector Y was defined in equation 4.1:

$$\vec{X} \cdot \vec{Y} = \sum_{i=1}^{i=n} X_i Y_i = X_1 Y_1 + X_2 Y_2 + \dots + X_n Y_n \quad (4.1)$$

Where X_n and Y_n are the frequency of words in both vectors where all terms within the news vectors have a frequency of value one and n is the dimension of the shortest vector, and it was the news vector in all cases. Figures; 4.6.a-4.6.f shows the dot product-process of web page vector with all news categories vectors.

```

DOT PRODUCT BETWEEN WEBPAGE VECTOR AND KEYWORDS VECTOR FOR BUSINESS NEWS
WORD report = 1 * 1 = 1
WORD news = 1 * 1 = 1
WORD plan = 1 * 1 = 1
WORD manag = 1 * 1 = 1
WORD new = 1 * 1 = 1
WORD navig = 1 * 1 = 1
WORD world = 1 * 1 = 1
WORD respons = 1 * 1 = 1

```

```

DOT PRODUCT ==>>8

```

```

=====

```

(a): Dot Product between Web Page Vector and Business News Vector

```

DOT PRODUCT BETWEEN WEBPAGE VECTOR AND KEYWORDS VECTOR FOR CRIME NEWS
WORD report = 1 * 1 = 1
WORD news = 1 * 1 = 1
WORD court = 1 * 1 = 1
WORD assault = 1 * 1 = 1
WORD shot = 1 * 1 = 1
WORD law = 1 * 1 = 1
WORD polici = 1 * 1 = 1
WORD stori = 1 * 1 = 1
WORD search = 1 * 2 = 2
WORD life = 1 * 1 = 1
WORD podcast = 1 * 1 = 1
WORD man = 1 * 1 = 1
WORD charg = 1 * 1 = 1

```

DOT PRODUCT ==>>14

(b): Dot Product between Web Page Vector and Crime News Vector

```

DOT PRODUCT BETWEEN WEBPAGE VECTOR AND KEYWORDS VECTOR FOR HEALTH NEWS
WORD news = 1 * 1 = 1
WORD help = 1 * 1 = 1
WORD report = 1 * 1 = 1
WORD new = 1 * 1 = 1
WORD top = 1 * 1 = 1
WORD world = 1 * 1 = 1
WORD life = 1 * 1 = 1

```

DOT PRODUCT ==>>7

(c): Dot Product between Web Page Vector and Health News Vector

```

DOT PRODUCT BETWEEN WEBPAGE VECTOR AND KEYWORDS VECTOR FOR POLITICS NEWS
WORD report = 1 * 1 = 1
WORD presid = 1 * 1 = 1
WORD polici = 1 * 1 = 1
WORD news = 1 * 1 = 1
WORD stori = 1 * 1 = 1
WORD american = 1 * 1 = 1
WORD world = 1 * 1 = 1

```

DOT PRODUCT ==>>7

(d): Dot Product between Web Page Vector and Politics News Vector

```

DOT PRODUCT BETWEEN WEBPAGE VECTOR AND KEYWORDS VECTOR FOR SPORTS NEWS
WORD team = 1 * 1 = 1
WORD game = 1 * 2 = 2
WORD sport = 1 * 2 = 2
WORD playeR = 1 * 2 = 2
WORD report = 1 * 1 = 1
WORD win = 1 * 2 = 2
WORD score = 1 * 1 = 1
WORD leagu = 1 * 1 = 1
WORD make = 1 * 1 = 1
WORD open = 1 * 1 = 1
WORD news = 1 * 1 = 1
WORD live = 1 * 1 = 1
WORD play = 1 * 1 = 1
WORD time = 1 * 1 = 1
WORD show = 1 * 1 = 1
WORD football = 1 * 1 = 1
WORD final = 1 * 2 = 2
WORD season = 1 * 1 = 1
WORD basketbal = 1 * 1 = 1
WORD box = 1 * 1 = 1
WORD tenni = 1 * 1 = 1
WORD golf = 1 * 1 = 1
WORD athlet = 1 * 1 = 1
WORD swim = 1 * 1 = 1
WORD coach = 1 * 1 = 1
WORD retir = 1 * 1 = 1
WORD race = 1 * 1 = 1
WORD champion = 1 * 2 = 2
WORD match = 1 * 1 = 1
WORD cricket = 1 * 1 = 1
WORD beat = 1 * 2 = 2
WORD olymp = 1 * 2 = 2
WORD cup = 1 * 1 = 1
WORD cycl = 1 * 1 = 1
WORD challeng = 1 * 1 = 1
WORD rugby = 1 * 1 = 1
WORD winner = 1 * 1 = 1

```

DOT PRODUCT ==>>45

(e): Dot Product between Web Page Vector and Sports News Vector

```

DOT PRODUCT BETWEEN WEBPAGE VECTOR AND KEYWORDS VECTOR FOR TECHNOLOGY NEWS
WORD report = 1 * 1 = 1
WORD game = 1 * 2 = 2
WORD featur = 1 * 2 = 2
WORD news = 1 * 1 = 1
WORD site = 1 * 1 = 1
WORD phone = 1 * 1 = 1
WORD chang = 1 * 1 = 1
WORD secur = 1 * 1 = 1
WORD websit = 1 * 1 = 1
WORD plan = 1 * 1 = 1
WORD stori = 1 * 1 = 1
WORD intern = 1 * 1 = 1
WORD media = 1 * 1 = 1
DOT PRODUCT ==>>15

```

(f): Dot Product between Web Page Vector and Technology News Vector

Figure 4.6: Dot Product Between Web Page Vector and News Categories Vectors

3. Calculating the length (or magnitude) of both, the web page vector and the news categories vectors using the formula in equation 4.2:

$$\|\vec{X}\| = \sqrt{\sum_{i=1}^{i=n} (X_i)^2} \quad \|\vec{Y}\| = \sqrt{\sum_{i=1}^{i=n} (Y_i)^2} \quad (4.2)$$

where n is the number of words in the vector and X_i and Y_i are the frequency of words in both vectors.

4. Calculating the cosine similarity through dividing the dot product of two vectors by the product of the two vectors lengths, as the formula in equation 4.3:

$$\text{Similarity}_{\text{web page, news vector}} = \frac{\vec{X} \cdot \vec{Y}}{\|\vec{X}\| \cdot \|\vec{Y}\|} \quad (4.3)$$

5. Steps (2)-(4) were repeated between the web page vector and each one of the six news vectors.
6. Subsequent to the achievement of computing cosine similarity between the web page vector and news vectors, the web page was categorised and added to the category that held the highest cosine similarity value, which was, in this case, the sports category and its similarity value (0.324101861776082). Figure 4.7 displays the results of computing cosine similarity of (<http://www.bbc.com/sport/tennis>) web page.
7. Calculating the average of cosine similarity value between the web page similarity value and the similarity of web page URL anchor text (this will be explained later in class7).

$$\text{Similarity}_{(\text{page, sport news category})} = \text{Cosine } \Theta_{(\text{page, sport news category})} =$$

$$\frac{0.324101861776082 + 0.14002800840280097}{2} = 0.232064935089441485$$

8. Comparing the similarity value with the sports category threshold value. In case the similarity value was greater than or equal the threshold value, the web page would be accepted, or otherwise ignored.

```

BUSINESS SIMILARITY ==>>
DOT PRODUCT ==>> 8
LENGTH PRODUCT OF WEB PAGE VECTOR ==>> 19.4422220952236
LENGTH PRODUCT OF NEWS VECTOR ==>> 6.32455532033676
CONSINE SIMILARITY OF BUSINESS NEWS CATEGORY ==>> 0.0650600048632355
=====

CRIME SIMILARITY ==>>
DOT PRODUCT ==>> 14
LENGTH PRODUCT OF WEB PAGE VECTOR ==>> 19.4422220952236
LENGTH PRODUCT OF NEWS VECTOR ==>> 8.60232526704263
CONSINE SIMILARITY OF CRIME NEWS CATEGORY ==>> 0.0837078670556538
=====

HEALTH SIMILARITY ==>>
DOT PRODUCT ==>> 7
LENGTH PRODUCT OF WEB PAGE VECTOR ==>> 19.4422220952236
LENGTH PRODUCT OF NEWS VECTOR ==>> 5.91607978309962
CONSINE SIMILARITY OF HEALTH NEWS CATEGORY ==>> 0.0608580619450185
=====

POLITICS SIMILARITY ==>>
DOT PRODUCT ==>> 7
LENGTH PRODUCT OF WEB PAGE VECTOR ==>> 19.4422220952236
LENGTH PRODUCT OF NEWS VECTOR ==>> 5.56776436283002
CONSINE SIMILARITY OF POLITICS NEWS CATEGORY ==>> 0.0646652994719309
=====

SPORT SIMILARITY ==>>
DOT PRODUCT ==>> 45
LENGTH PRODUCT OF WEB PAGE VECTOR ==>> 19.4422220952236
LENGTH PRODUCT OF NEWS VECTOR ==>> 7.14142842854285
CONSINE SIMILARITY OF SPORT NEWS CATEGORY ==>> 0.324101861776082
=====

TECHNOLOGY SIMILARITY ==>>
DOT PRODUCT ==>> 15
LENGTH PRODUCT OF WEB PAGE VECTOR ==>> 19.4422220952236
LENGTH PRODUCT OF WEB KEYWORDS VECTOR ==>> 8.06225774829855
CONSINE SIMILARITY OF TECHNOLOGY NEWS CATEGORY ==>> 0.0956948752938691
=====

WEB PAGE CATEGORY =====>> SPORTS

```

Figure 4.7: Results of Computing Cosine Similarity of (<http://www.bbc.com/sport/tennis>) Web Page

The results of this class illustrated the similarity value and the category of the web page.

4.2.6 Class 6: Reading Robot File

The purpose of this class is to read and extract disallowed (blocked) URLs of a website that is not visited by the crawler. In addition, at the beginning of each crawling process, a temporary table named (Blocked_URLs Table) was created and used to save all the blocked URLs of all the crawled websites. When the crawling process terminated, the table would be deleted, that is due to the modifications that may have happened to the robot file during the crawling processes. Likewise, the idea of designing the table was to decrease the network loading and time consumed in downloading the corresponding file every time a web page of the same website was being crawled. However, when the crawler visited, the webpage would check whether the blocked URLs of the website,

to which the web page belonged, have been saved or not. In case they have been stored, the blocked URLs were retrieved, and if they were not following the steps, they were executed.

By using (<http://www.bbc.com/sport/tennis>), the following steps demonstrate how to read and process the robot file:

1. Create a robot file URL, through extracting the URL domain name, including the URL schema, authority, and an appendix by the path (“/robots.txt”), so the URLs would appear like (<http://www.bbc.com/robots.txt>).
2. Download the file and convert it into a text string in order to be processed by implementing class1. Figure 4.8 displays a portion of www.bbc.com robot file text string.

```
# www.bbc.com
User-agent: Googlebot
Sitemap: http://www.bbc.com/sitemap.xml
Sitemap: http://www.bbc.com/sitemaps/index-com-news.xml
Sitemap: http://www.bbc.com/sitemaps/index-com-archive.xml
Sitemap: http://www.bbc.com/video_sitemap.xml
Disallow: /iplayer/episode/*?from=r*
Disallow: /iplayer/cy/episode/*?from=r*
Disallow: /iplayer/gd/episode/*?from=r*
Disallow: /_programmes
Disallow: /606/
Disallow: /aboutthebbc/insidethebbc/search
```

Figure 4.8: A Portion of (www.bbc.com) Robot File Text String

3. Part the text into a collection of lines, where each line treated individually. As to the BBC robot file, which contains 656 line.
4. The lines would be separated into two portions (two string arrays) based on the colons (:) where the first part is named (command) and the second is (URL) as shown in the example in Figure 4.9. Also, comment lines which begin with a hashtag (#) will be disregarded.


Disallow: /_programmes


Figure 4.9: Separating the Line of the Robot File string

5. Processing the robot file user-agent and disallow commands. Allow and sitemap was ignored in this dissertation:
 - a. User-agent Command: when processing a user agent command, the crawler checked whether the second part (URL part) was a star (*) that indicates all user agents including our user agent which do not match other user agent names mentioned in the file. In this case, the command would be processed and otherwise neglected. Additionally, in (www.bbc.com) robot file,

several user agents like Googlebot, msnbot, Googlebot-Mobile were mentioned along with their blocked URLs. Figure E.1.a in Appendix (E) shows a portion of blocked URLs for Googlebot agent and Figure E.1.b in Appendix (E) displays a collection of blocked URLs to all agents.

b. Disallow Command: after detecting (user-agent:*) command, disallowed commands following the user agent would be immediately processed and an empty list called (Blocked-list) will be created for the purpose of adding blocked URLs to it. For this command, the next URL to disallow word would be severed and added to the Blocked list, where a URL which includes this portion of URL should not be retrieved by the crawler later. Figure E.2.a and Figure E.2.b in Appendix (E) exhibit blocked URLs of Googlebot agent and other agents respectively.

6. After finishing processing the disallow commands, the Blocked-list contained 145 URL as shown in Figure E.3.a in Appendix (E) were filled into the (Blocked_URLs Table) along with their site URL (*www.bbc.com*) to be regained later to all URLs that belonged to BBC site. Figure E.3.b in Appendix (E) shows the blocked URL extracting of (*http://www.bbc.com/news/topics/c349zr6w3zjt/republican-party*) web page after detecting changes that happened in the web page (Detecting changes will be discussed in details later in section 4.2.8). The URL contained (news round) blocked path which was extracted from robot file previously and stored in the Blocked-URLs Table.

The outcome of this class was a list of blocked URLs of the websites.

4.2.7 Class 7: Extracting URLs of Web Pages

In this class, URLs and their anchor text were extracted, normalised and attached to the frontier. URLs that were images or files were neglected and URL marked with a hashtag would be ignored too because they led to retrieving the same URL. The following steps demonstrate how to normalise and prioritise URLs, while continuing to use (*http://www.bbc.com/sport/tennis*), which is considered as the base URL:

1. Text string of web page would be retrieved and unnecessary tags were removed except HTML tags using class 2.
2. Extracting URLs of the web page text string, through searching for anchor tags to extract their href attribute values and anchor text in order to be used in:

- a. Managing URLs in the Frontier, through assigning downloading priorities to URLs in the frontier.
 - b. Computing cosine similarity of the web page (as discussed in section 4.2.5).
3. Canonicalisation of URLs, this step involves two cases:
- a. Case1: absolute URLs: it's URLs which includes the full path of the page. Because we needed to restrict the domain to the seed sites (BBC, Aljazeera, TheGuardian, and ABC news) (will be discussed in section 4.2.9); therefore, when the crawler caught an absolute URL, it checked its domain name and verified whether it related to the domain of these sites or not. Figure 4.10.a shows an examples of accepted absolute URLs and Figure 4.10.b examples of ignored URLs.

```

=====
URL: http://www.bbc.com/earth/
URL ANCHOR TEXT: Earth
** ABSOLUTE URL **
NORMALIZED URL: http://www.bbc.com/earth/
COSINE SIMILARITY : 0
CATEGORY          : NOT CATEGORIZED
-----
URL: http://www.bbc.com/travel/
URL ANCHOR TEXT: Travel
** ABSOLUTE URL **
NORMALIZED URL: http://www.bbc.com/travel/
COSINE SIMILARITY : 0
CATEGORY          : NOT CATEGORIZED
-----
URL: http://www.bbc.com/capital/
URL ANCHOR TEXT: Capital
** ABSOLUTE URL **
NORMALIZED URL: http://www.bbc.com/capital/
COSINE SIMILARITY : 0
CATEGORY          : NOT CATEGORIZED
-----
(a): Examples of Absolute URLs that refer to the Current Page Domain
=====

URL: http://www.lta.org.uk/
URL ANCHOR TEXT: LTA
** ABSOLUTE URL **
ABSOLUTE URL BUT DOES NOT BELONG TO THE CURRENT PAGE DOMAIN
-----
URL: http://www.uksport.gov.uk/
URL ANCHOR TEXT: UK Sport
** ABSOLUTE URL **
ABSOLUTE URL BUT DOES NOT BELONG TO THE CURRENT PAGE DOMAIN
=====
(b): Examples of Absolute URLs that did not Refer to the Current Page Domain

```

Figure 4.10: Absolute URLs Extracted of (<http://www.bbc.com/sport/tennis>) Web Page

- b. Case2: relative URLs: there were two forms of relative URLs, root relative URLs, and dot-dot-slash URLs. The following explanation shows how they will be processed:
 - i. Root Relative URLs: URLs began with a slash“/” where this fore slash means that these URLs were relative to the root URL (URL of the site), not the current page path. In addition, normalising this kind of URLs would be achieved by combining site’s URL into the relative URL. Figure 4.11.a gives an illustration of how to normalise root relative URL and Figure 4.11.b gives a specimen of our results.

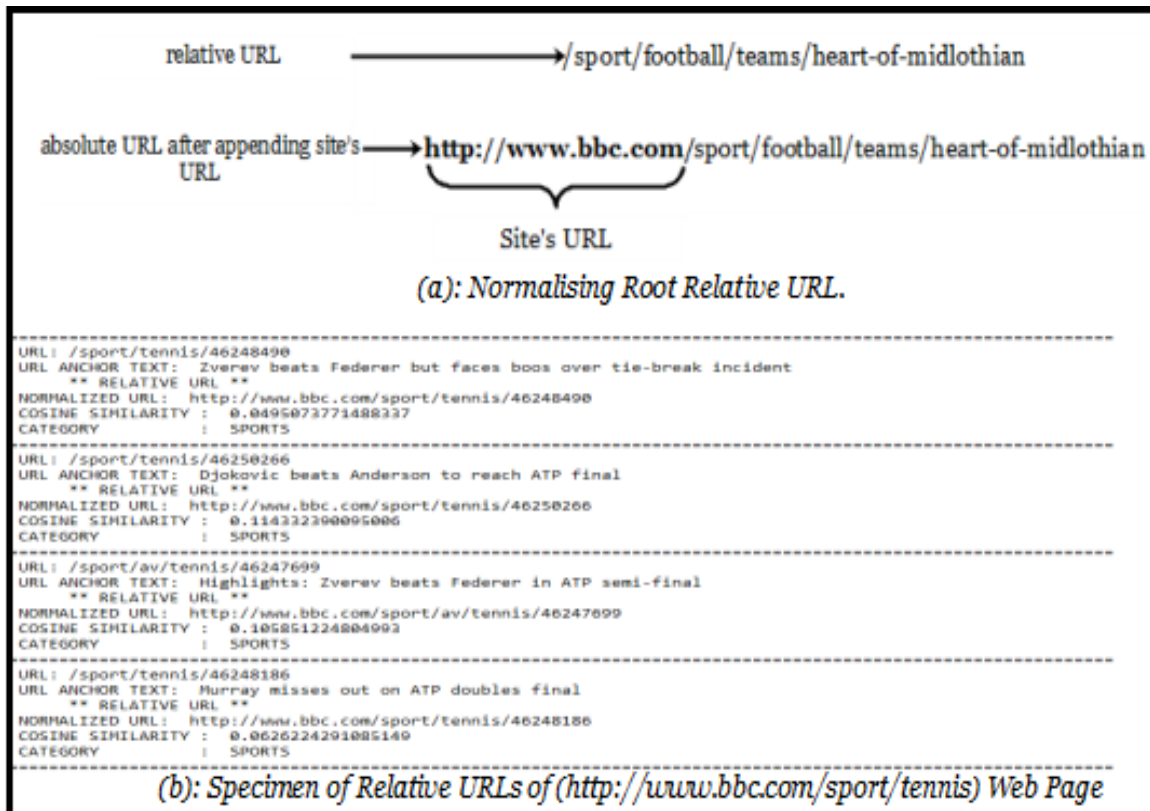


Figure 4.11: Root Relative URLs Extracted of (<http://www.bbc.com/sport/tennis>) Web Page

- ii. Dot-dot-slash (“../”) URLs: this was a different kind of relative URLs using dot-dot-slashes, where every dot-dot-slash ordered the crawler to move back one folder position. In order to process this kind of URLs, the crawler eliminated all dot-dot-slashes, where every elimination process of (“../”) matched an elimination of a folder. After completing, the base URL was prefixed to the relative URL. Figure 4.12 illustrates how to normalise dot-dot-slashes URLs.

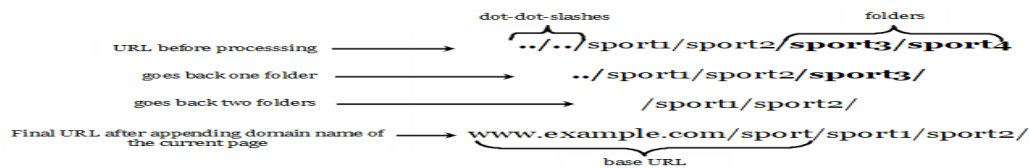


Figure 4.12: Normalising Dot-Dot-Slash (“../”) URLs

4. Calculate the similarity of URL anchor text, through implementing the following steps:
 - a. Creating Anchor Text Vector: to create anchor text vector, class 2 was implemented and since anchor text extracted and did not need to download, only tidying steps which were tokenising,

eliminating stop words, performing the stemming process and creating a vector of terms with their frequencies were executed.

b. Calculating Cosine Similarity: similarity value between anchor text and categories vectors would be calculated by implementing class 5 but rather than web page vector the class would receive anchor text vector and calculating its similarity. The previously displayed Figure 4.11.b shows URLs anchor texts and their similarity values.

5. Attaching URLs to the frontier, in which every extracted URL, that has not been added earlier to the frontier, would be added to the frontier along with its similarity value.

Results of this class displayed a list of extracted and allowed URLs and the cosine similarity of their anchor text.

4.2.8 Class 8: Detecting Changes of Web pages

In this class, the crawler revisited the web pages in order to detect their changes. To begin with, the crawler would go through all the news categories tables (Business, Crime, Health, Politics, Sports and Technology) one by one, extract their URLs, crawl them again and detect changes that happened in them. Before starting the crawling process, the threshold similarity value has to be defined for each news category, through implementing Class 4 (the second case). The following procedures have to be carried out before identifying changes in web pages (For more about these procedures turn back to chapter 3, section 3.2):

1. URLs would be retrieved from the database tables and added into the frontier in descending order (this step will be discussed later in section 4.2.9). And URLs would be extracted one by one from the frontier.
2. The new web page would be downloaded and the old web page retrieved from the archive. Likewise; their trees would be built through implementing class 1 without removing HTML tags since it would be used in detecting the changes.
3. Web page category would be verified to see if it has changed or not and check whether web page was accepted in the new category or not. Figure 4.13.a and Figure 4.13.b show an instance of web pages their changed category.

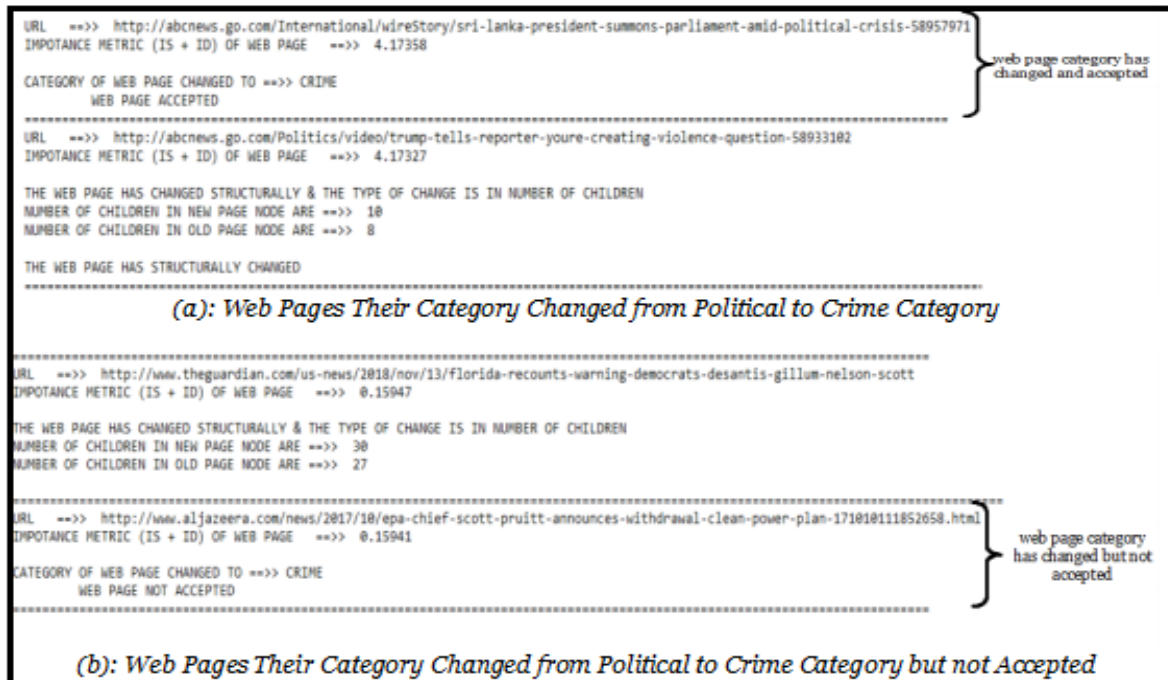


Figure 4.13: Recategorising Web Pages

There are two kinds of changes that will be considered in this dissertation, and they are the structural and textual changes.

4.2.8.1 Structural Changes

In this kind of changes, the crawler would recursively traverses the old and new trees, and each old node would be compared with its corresponding new node. In case the type of both nodes was not identical, or a number of children of both nodes was not equal, then, the detecting changes process would be stopped and URLs' structural counter will increase by one, web pages would be replaced, the robot file would be read and URLs would be extracted implementing class 6 and 7 respectively. Figure 4.14 shows business web pages that have been structurally changed.

```

-----
URL ==> http://www.bbc.com/news/business-45937741
IMPOTANCE METRIC (IS + ID) OF WEB PAGE ==> 5.20108

THE WEB PAGE HAS CHANGED STRUCTURALLY & THE TYPE OF CHANGE IS IN NUMBER OF CHILDREN
NUMBER OF CHILDREN IN NEW PAGE NODE ARE ==> 2
NUMBER OF CHILDREN IN OLD PAGE NODE ARE ==> 0

THE WEB PAGE HAS STRUCTURALLY CHANGED
-----
URL ==> http://www.bbc.com/news/world-us-canada-44234245
IMPOTANCE METRIC (IS + ID) OF WEB PAGE ==> 5.19475

THE WEB PAGE HAS CHANGED STRUCTURALLY & THE TYPE OF CHANGE IS IN NODE TYPE
NODE TYPE IN NEW PAGE NODE ARE ==> a
NODE TYPE IN OLD PAGE NODE ARE ==> div

THE WEB PAGE HAS STRUCTURALLY CHANGED
-----
URL ==> http://www.bbc.com/news/business-44706880
IMPOTANCE METRIC (IS + ID) OF WEB PAGE ==> 5.19429

THE WEB PAGE HAS CHANGED STRUCTURALLY & THE TYPE OF CHANGE IS IN NUMBER OF CHILDREN
NUMBER OF CHILDREN IN NEW PAGE NODE ARE ==> 2
NUMBER OF CHILDREN IN OLD PAGE NODE ARE ==> 0

THE WEB PAGE HAS STRUCTURALLY CHANGED
-----
URL ==> http://www.theguardian.com/business-to-business
IMPOTANCE METRIC (IS + ID) OF WEB PAGE ==> 5.19241

THE WEB PAGE HAS CHANGED STRUCTURALLY & THE TYPE OF CHANGE IS IN NUMBER OF CHILDREN
NUMBER OF CHILDREN IN NEW PAGE NODE ARE ==> 0
NUMBER OF CHILDREN IN OLD PAGE NODE ARE ==> 1

THE WEB PAGE HAS STRUCTURALLY CHANGED
-----

```

Figure 4.14: Web pages of Business Category had Structurally Changed

In the process of detecting structural changes, the crawler would keep traversing both trees and comparing their nodes until it detected changes or finished scanning all the nodes. However, if there were not any structural changes, the crawler would verify if there are any textual changes.

4.2.8.2 Textual changes

Detecting textual changes is considered of two stages:

1. Identifying nodes whose texts have changed.
2. Checking the text of nodes that have been previously changed.

The web page (<http://www.bbc.com/news/correspondents/davelee>) will be used as an example to demonstrate these caeses:

1. Case 1: case web page have no previously stored textual parts, in this process the crawler would recognise textual nodes whose texts have changed, detecting their path, extracting their text and saving them into database tables as a dynamic nodes of the page. For more about this process, return back to chapter 3 section 3.2.1.2. Figure 4.15 displays the textual changes appeared in the case page, followed by an explanation of these changes.

```

=====
URL ==>> http://www.bbc.com/news/correspondents/davelee
IMPOTANCE METRIC (IS + ID) OF WEB PAGE ==>> 0.17851

TEXTUAL CHNAGE:
NODE TEXT IN NEW PAGE ==>> Is Facebook a friend to local journalism?
NODE PATH IN NEW PAGE ==>> /html[1]/body[1]/div[1]/div[6]/div[1]/div[4]/div[2]/div[1]/div[2]/div[1]/div[1]/div[1]/div[1]

NODE TEXT IN OLD PAGE ==>> Tech Tent: Has Facebook reached peak crisis?
NODE PATH IN OLD PAGE ==>> /html[1]/body[1]/div[1]/div[6]/div[1]/div[4]/div[2]/div[1]/div[2]/div[1]/div[1]/div[1]/div[1]/a[1]/h2[1]/#text[1]
*****

TEXTUAL CHNAGE:
NODE TEXT IN NEW PAGE ==>> Facebook has unveiled a scheme to train reporters to work in local communities across the UK.
NODE PATH IN NEW PAGE ==>> /html[1]/body[1]/div[1]/div[6]/div[1]/div[4]/div[2]/div[1]/div[2]/div[1]/div[1]/div[1]/div[1]

NODE TEXT IN OLD PAGE ==>> Tech Tent looks at the latest scandal to hit Facebook and a looming privacy debate for Google.
NODE PATH IN OLD PAGE ==>> /html[1]/body[1]/div[1]/div[6]/div[1]/div[4]/div[2]/div[1]/div[2]/div[1]/div[1]/div[1]/div[1]/p[1]/#text[1]
*****

THE WEB PAGE HAS TEXTUALLY CHANGED
=====

```

Figure 4.15: Textual Changes in (<http://www.bbc.com/news/correspondents/davelee>) Web Page

As it is shown in Figure 4.22 two textual changes happened in the page. The first textual change was recognised where the text of the old tree “Tech Tent: Has Facebook reached peak crisis?”. Also, in the new tree node “Is Facebook a friend to local journalism?”. The node path in the new tree:

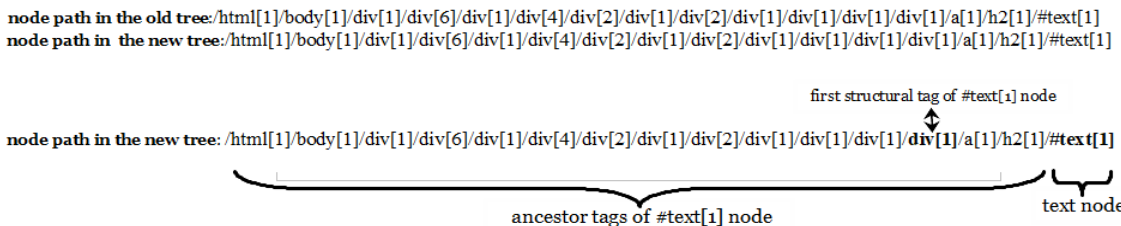
`/html[1]/body[1]/div[1]/div[6]/div[1]/div[4]/div[2]/div[1]/div[2]/div[1]/div[1]/div[1]/div[1]/div[1]/a[1]/h2[1]/#text[1]`. And the node path in the old tree was as:

`/html[1]/body[1]/div[1]/div[6]/div[1]/div[4]/div[2]/div[1]/div[2]/div[1]/div[1]/div[1]/div[1]/div[1]/a[1]/h2[1]/#text[1]`.

Before saving the path, the crawler had to look backward for the first structural ancestor tag of (`#text[1]`). Then, the node skipped any tags after it and added the rest of the path to a temporary table T (`#text[1]`). In this case, `div[1]` was the first ancestor tag of the node. Therefore, the path would be saved as:

`/html[1]/body[1]/div[1]/div[6]/div[1]/div[4]/div[2]/div[1]/div[2]/div[1]/div[1]/div[1]/div[1]`.

Figure 4.16 illustrates the process of handling the path.



node path in the new tree: `/html[1]/body[1]/div[1]/div[6]/div[1]/div[4]/div[2]/div[1]/div[2]/div[1]/div[1]/div[1]`

Figure 4.16: The Process of Handling Path of Text Node

The second textual change, where the old text was “Tech Tent looks at the latest scandal to hit Facebook and a looming privacy debate for Google”, and the new text “Facebook has unveiled a scheme to train reporters to work in local communities across the UK” . As for the paths, the node path in the new tree was:

/html[1]/body[1]/div[1]/div[6]/div[1]/div[4]/div[2]/div[1]/div[2]/div[1]/div[1]/div[1]/div[1]/p[1]/#text[1]. And the node path in the old tree:

/html[1]/body[1]/div[1]/div[6]/div[1]/div[4]/div[2]/div[1]/div[2]/div[1]/div[1]/div[1]/div[1]/p[1]/#text[1].

This time when processing the path in order to determine the first structural ancestor tag of (#text[1]) node, it was (div[1]) and the path was:

/html[1]/body[1]/div[1]/div[6]/div[1]/div[4]/div[2]/div[1]/div[2]/div[1]/div[1]/div[1]/div[1],

which means that it corresponds with the previously stored path. Therefore, this path would be neglected and would not be added to the table.

As noted in the first and second changes, both #text nodes had the same parent structural node div[1] see Figure 4.17. Thus, this node would be considered as the dynamic node, whose text has changed. Also, its text: “Is Facebook a friend to local journalism? Facebook has unveiled a scheme to train reporters to work in local communities across the UK.”.

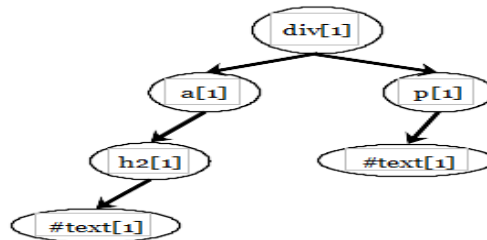


Figure 4.17: Paths of #Text[1] Nodes

After finishing checking the text of web page nodes, the temporary table T which contained only one node and its data text, path and its counter which would be created and initiated to 1 would be attached to the technology dynamic parts table.

2. Case 2: checking the text of nodes that have been previously changed and has the highest number of changes: When the crawler revisited the web page (<http://www.bbc.com/news/correspondents/davelee>), it had to ensure that the web page did not had any structural changes, in or-

der for the crawler to begin figuring out textual changes, and since the web page had previously stored textual parts (dynamic parts), then:

- a. Webpage path and its text would be retrieved. The path:
`(/html[1]/body[1]/div[1]/div[6]/div[1]/div[4]/div[2]/div[1]/div[2]/div[1]/div[1]/div[1]/div[1])` and its text “Is Facebook a friend to local journalism? Facebook has unveiled a scheme to train reporters to work in local communities across the UK.”
- b. The text of the corresponding path in the new tree will be extracted and compared with the old text. The text in the new page was “Tech Tent: Hot gadgets, self-driving cars and a weird party at CES From a smart plank of wood to a self-driving Russian car, there's endless fun at the Las Vegas trade show.” since the text of the node changed, its counter will be incremented by one and its text in database table will be replaced also. Figure 4.18 demonstrates the result of the comparing process and Figure 4.19 presents the dynamic nodes in the web page.
- c. Every time the crawler detecting textual changes in this page, the path of this node will be retrieved and its text will be checked. But in the other case, if the page has many dynamic nodes only the nodes that have the highest counter will be retrieved.

After detecting textual changes either by the first or second cases, the textual counter of the web page will be incremented by 1, and the web pages will be replaced. And the proceeding actions should be implemented the robot file should be read and URLs has to be extracted through implementing class 6 and 7 respectively.

```
=====
URL ==> http://www.bbc.com/news/correspondents/davelee
IMPORTANCE METRIC (IS+ID) OF WEB PAGE ==> 0.17851

TEXTUAL CHANGE IN WEB PAGE DYNAMIC PARTS
NODE TEXT IN OLD PAGE ==> Is Facebook a friend to local journalism? Facebook has unveiled a scheme to train reporters to work in local communities across the UK.
NODE PATH IN OLD PAGE ==> /html[1]/body[1]/div[1]/div[6]/div[1]/div[4]/div[2]/div[1]/div[2]/div[1]/div[1]/div[1]/div[1]

NODE TEXT IN NEW PAGE ==> Tech Tent: Hot gadgets, self-driving cars and a weird party at CES From a smart plank of wood to a self-driving Russian car, there's endless fun at the Las Vegas trade show.
NODE PATH IN NEW PAGE ==> /html[1]/body[1]/div[1]/div[6]/div[1]/div[4]/div[2]/div[1]/div[2]/div[1]/div[1]/div[1]/div[1]
*****

THE WEB PAGE HAS TEXTUALLY CHANGED
=====
```

Figure 4.18: Changes in Textual Dynamic Nodes of (<http://www.bbc.com/news/correspondents/davelee>) Web Page



Figure 4.19: Textual Dynamic Part in (<http://www.bbc.com/news/correspondents/davelee>) Web Page

4.2.9 Class 9: The Frontier

It is a class concentrated on receiving and ordering URLs using two importance metrics similarity values and the changes happened in their web pages for the purpose of passing them to the crawler to prioritise their pages. There were two cases regarding the selection of URLs to supply the frontier with. The first case was when the crawler initialises for the first time, and the other case was when it crawled the earlier stored URLs in the database table:

1. The First Case: Initialising the frontier with seed URLs which were provided by the user.

In this status, the crawler started with seed URLs that were provided by the user and crawled their pages, extracting and going after their links and crawling their web pages also. This process of crawling web pages and extracting their URLs would be continued until the crawler visited a particular number of pages (Siqueira et al., 2017). However, in this experiment, the used seed URLs were the URL of the homepages of the popular News websites, particularly BBC, Al-Jazeera, The Guardian, and ABC news and the steps of this method perform as follows:

- a. Creating URLs queue, which is a list that handles objects as pairs of keys and their values, where the keys represent URLs and the values represent their importance metric, which in this case, their cosine similarity values.
- b. Initialising URLs queue by seed URLs and setting their similarity values to one. The reason behind that is because if their values initialise to zero, the crawler will go in depth of the first URL and may never reach the other seed URLs. The used seed URLs are:
 - i. <http://www.bbc.com/>
 - ii. <http://www.aljazeera.com/>
 - iii. <http://www.theguardian.com/international>
 - iv. <http://abcnews.go.com/>
- b. Categorising seed URLs, URLs pages would be visited, processed and classified by classes 1,2, and 5 respectively. As homepages of news websites had various classes of news (business, politics, sport, ..., etc); therefore, they did not have a distinct category. Hence, they were classified and their URLs were extracted and inserted to the frontier without attaching them to the database tables. Figure 4.20 illustrates the results of crawling and classifying seed pages.

```

=====
URL:      http://www.bbc.com/
SIMILARITY METRIC OF URL ANCHOR TEXT ==>> 1
SIMILARITY METRIC OF WEB PAGE TEXT   ==>> 0.12087497762613
AVERAGE OF SIMILARITY METRIC (IS)    ==>> 0.560437488813065
WEB PAGE CATEGORY                      ==>> SPORTS
=====

URL:      http://www.aljazeera.com/
SIMILARITY METRIC OF URL ANCHOR TEXT ==>> 1
SIMILARITY METRIC OF WEB PAGE TEXT   ==>> 0.21187283673032
AVERAGE OF SIMILARITY METRIC (IS)    ==>> 0.60593641836516
WEB PAGE CATEGORY                      ==>> CRIME
=====

URL:      http://www.theguardian.com/international
SIMILARITY METRIC OF URL ANCHOR TEXT ==>> 1
SIMILARITY METRIC OF WEB PAGE TEXT   ==>> 0.14554022002581
AVERAGE OF SIMILARITY METRIC (IS)    ==>> 0.572770110012905
WEB PAGE CATEGORY                      ==>> BUSINESS
=====

URL:      http://abcnews.go.com/
SIMILARITY METRIC OF URL ANCHOR TEXT ==>> 1
SIMILARITY METRIC OF WEB PAGE TEXT   ==>> 0.164185343306968
AVERAGE OF SIMILARITY METRIC (IS)    ==>> 0.582092671653484
WEB PAGE CATEGORY                      ==>> CRIME
=====

```

Figure 4.20: The Results of Crawling and Classifying Seed URLs Pages

- c. Start in the processing of extracting URLs from the frontier crawling, classifying their web pages and inserting them into their news category tables in the database. This process contin-

ued 4307 times in this experiment, while we were initialising the frontier for the first crawling process. Figure 4.21 shows an example of crawled URLs in the first crawling process.

```

=====
URL: http://www.theguardian.com/business/2018/sep/23/trump-trade-war-china-beijing-tariffs-technology
SIMILARITY METRIC OF URL ANCHOR TEXT ==> 0.15
SIMILARITY METRIC OF WEB PAGE TEXT ==> 0.179398740368981
AVERAGE OF SIMILARITY METRIC (IS) ==> 0.16469937018449
WEB PAGE CATEGORY ==> BUSINESS
WEB PAGE ACCEPTED AND STORED

=====
URL: http://www.theguardian.com/world/video/2018/jun/10/the-gig-is-up-trump-issues-warning-on-trade-at-g7-summit-video
SIMILARITY METRIC OF URL ANCHOR TEXT ==> 0.193649167310371
SIMILARITY METRIC OF WEB PAGE TEXT ==> 0.10285210532637
AVERAGE OF SIMILARITY METRIC (IS) ==> 0.14825063631837
WEB PAGE CATEGORY ==> POLITICS
WEB PAGE NOT ACCEPTED

=====
URL: http://www.theguardian.com/business/live/2018/oct/29/markets-edgy-ftse-financial-crisis-losses-china-business-live
SIMILARITY METRIC OF URL ANCHOR TEXT ==> 0.15
SIMILARITY METRIC OF WEB PAGE TEXT ==> 0.183315714445845
AVERAGE OF SIMILARITY METRIC (IS) ==> 0.166657857222923
WEB PAGE CATEGORY ==> BUSINESS
WEB PAGE ACCEPTED AND STORED

=====
URL: http://www.aljazeera.com/news/2018/10/germany-arrests-suspect-killing-viktoria-marinova-181010093524925.html
SIMILARITY METRIC OF URL ANCHOR TEXT ==> 0.149626400416145
SIMILARITY METRIC OF WEB PAGE TEXT ==> 0.229418852351581
AVERAGE OF SIMILARITY METRIC (IS) ==> 0.189522626383863
WEB PAGE CATEGORY ==> CRIME
WEB PAGE ACCEPTED AND STORED
=====

```

Figure 4.21: Examples of Crawled URLs in the First Cralwing Process

2. The Second Case: initialising the frontier with the beforehand stored URLs:

The crawler would get a list of URLs of the database table, starting with the business table next to crime, health, politics, sports and lastly technology news table. The Following procedures were executed to all tables one by one and the political table at the seventh crawling process would be practiced for illustration:

- a. URLs would be retrieved from the political table, which included 487 URLs that added them into a temporary table T.
- b. The crawler would create two frontiers: the static and dynamic frontier. As for the static queue, it included URLs' web pages that have never changed their metric, only their cosine similarity values which ranged between zero and one. However, in case of the dynamic queue contained URLs whose metrics were greater than or equal to one, they would be represented in cosine similarity and dynamicity values.
- c. All URLs within the temporary table T would be split between the static and dynamic queue, depending on their importance metrics values. The number of URLs for the dynamic and static queues was 356 and 122 URLs respectively. Figure F.1 in Appendix (F) demonstrates a set of political URLs in the static and dynamic queue at the crawling process number.

- d. The dynamic queue would be crawled first, followed by the static queue. While crawling URLs, any web page whether dynamic or static that changed, its URLs would be extracted to be attached to the static queue, and every time a URL was added to the static queue, the queue would be reordered based on the similarity value of its anchor text in descending order. Therefore, when the crawler started crawling the static queue, sometimes the new URLs that had a greater similarity than old URLs would be crawled first. Figure F.2 in Appendix (F) shows a process of crawling new URLs in which their similarity value of their anchors' texts were greater than the similarity value (anchor text and web page similarity value) of the previously stored URLs.
- e. URLs extracting of web pages while crawling URLs of the political news category, if any URL whose category was not political would be neglected, and the crawler would only concentrate on the current category.

These nine classes were operating in a sequence while the crawler was executing, as in following steps:

1. Step 01: Create terms vectors that represent news categories (class3).
2. Step 02: Determining similarity threshold values (class 4).
3. Step 03: Initializing frontier with seed URLs (Class9).
4. Step 04: Extracting URL of the frontier and fetching their corresponding web page (class1).
5. Step 05: Parsing web page to create its vector (class2).
6. Step 06: Calculating the web page cosine similarity (class 5). Just in case web page similarity value greater than or equal the defined threshold following steps will be executed otherwise, the next URL will be extracted of the frontier.
7. Step 07: Detecting changes in the web page, in case the web page is not crawling for the first time (class 8).
8. Step 08: Processing robot file (class 6).
9. Step 09: Extracting URLs of the web page (class 7).
10. Step 10: repeating step (2)_(9) until a specific number of URLs is reached.
11. Step 11: collecting new terms using un_relevant URLs (class 3).

4.3 Summary

During this chapter, web pages were crawled and classified into their news category by the cosine similarity measure and the changes appeared in them were recursively discovered by the crawler. And the results of classification and detecting changes processes were both combined and used as an importance metrics in prioritising URLs of web pages in the frontier.

Chapter Five

Results and Discussions

In this chapter, we will display the results of implementing the crawling process on web pages presented in chapter 3, and which was performed in this experiment eight times. This chapter will be formed as follows: Section 5.1 will provide the terms representing news categories; section 5.2 will present the number of URLs collected from the sites used as seed URLs; section 5.3 will illustrate the characteristics of the news categories; section 5.4 will show the results of implementing the dynamic and static frontier; and finally section 5.5 will show a brief comparison between the proposed approach in this dissertation and other crawling methods and finally section 5.6 summarise this chapter.

5.1 News Terms

As discussed earlier in chapter 4 section 4.2.3, for each category a set of terms representing them will be defined. The implementation of the two methods mentioned in chapter 4 was effective in collecting the terms. In the second method, we noticed that in the advanced crawling stages the same terms began to be repeated in all the classified web page groups. Hence, at the seventh and eighth stages, we settled on implementing this method, and to be content with the sufficient cumulative number of terms.

The number of terms resulting for the categories of business, crime, health, politics, sports, and technology was 37, 73, 35, 30, 51 and 63, respectively. Table G.1 in Appendix (G) displays the total terms for all the news categories.

5.2 Maximum Number of URLs for Seed Websites

There are two methods to limit web crawlers: 1) crawl depth, and 2) maximum number of URLs; and since we used the best first crawler which downloads web pages based on their similarity values computed between web pages and news categories, then to control our crawler, we used the second option: maximum number of URLs. For each one of the following websites (BBC, Aljazeera, The Guardian and ABC news) that were used as seeds, URLs were restricted to 2000 URLs.

The total number of collected URLs through the crawling process were 1994 URLs, distributed between websites, such as BBC, with a number of 1001 URLs, THE GUARDIAN with 706 URLs,

343 URLs for ALJAZEERA, and ABC NEWS with a total of 439 URLs.

5.3 Categories Characteristics

The results of crawling news categories were divided as follows:

1. Number of pages for each category.
2. Evaluation of web crawler for each category.
3. Total number of changes happening to the web pages of the categories at each crawling phase.
4. Total number of dynamic and static URLs for each category at each crawling phase.

5.3.1 Number of Pages for Each Category

A web crawler visits the web pages and categorised them by implementing the cosine similarity to one of the previously specified news categories. Table 5.1 shows the total number of URLs and their websites for each category in descending order, where the sports category has the highest number of URLs.

Table 5.1: Number of URLs for Each News Category

Category	Number of Pages	Website	Number of URLs
SPORTS	739	BBC	661
		ALJAZEERA	12
		THE GUARDIAN	64
		ABC NEWS	2
POLITICS	428	BBC	45
		ALJAZEERA	128
		THE GUARDIAN	189
		ABC NEWS	66
BUSINESS	404	BBC	211
		ALJAZEERA	8
		THE GUARDIAN	166
		ABC NEWS	19
HEALTH	278	BBC	74
		ALJAZEERA	1
		THE GUARDIAN	199
		ABC NEWS	4
CRIME	196	BBC	0
		ALJAZEERA	188

Category	Number of Pages	Website	Number of URLs
CRIME	196	THE GUARDIAN	0
		ABC NEWS	8
TECHNOLOGY	100	BBC	10
		ALJAZEERA	1
		THE GUARDIAN	87
		ABC NEWS	2

5.3.2 Focused Crawler Performance for Each Category

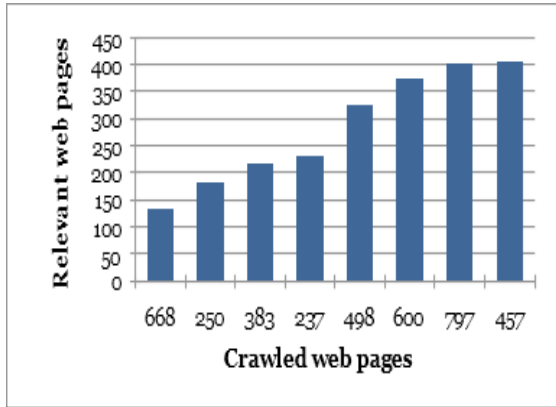
In this section, the performance of the crawler for each category will be evaluated by harvest ratio, is an important information retrieval performance metric, and widely used in evaluation the focused web crawler performance. To clarify, harvest ratio is used to measure the efficiency of the crawler in crawling relevant pages and to discard the irrelevant ones; its value ranges from zero to one. Harvest ratio is shown in equation 5.1 (Choudhary & Roy, 2013).

$$\text{Harvest ratio} = \frac{\text{Number of relevant pages}}{\text{Number of downloaded pages}} \quad (5.1)$$

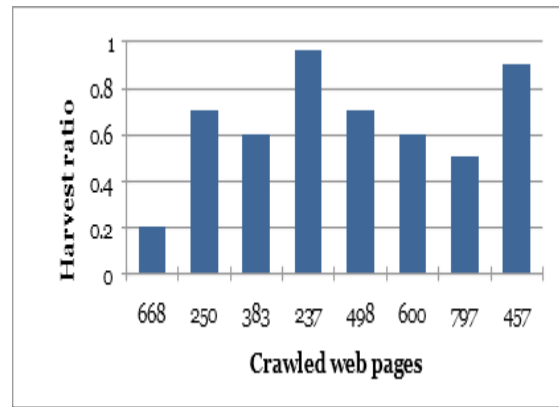
The effectiveness of a focused crawler is decided by employing the harvest ratio which measures the rate at which the relevant pages are crawled, and how effectively the irrelevant pages are eliminated from the crawling process.

The Figure 5.1 shows, for each category, the number of relative pages out of the total crawled ones, and the crawler's performance through all eighth phases. The maximum number of iterations for the crawler at each crawling phase was specified, but the number of pages crawled was rarely equal to the number of URLs specified. The reason for that is:

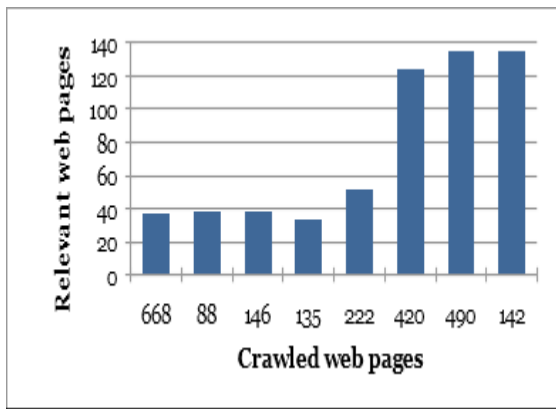
1. Most web pages have no HTTP response.
2. Due to the density of the links and their distribution inside the seed websites chosen for crime and technology categories, we can notice from the figure how few pages were crawled, unlike other categories.



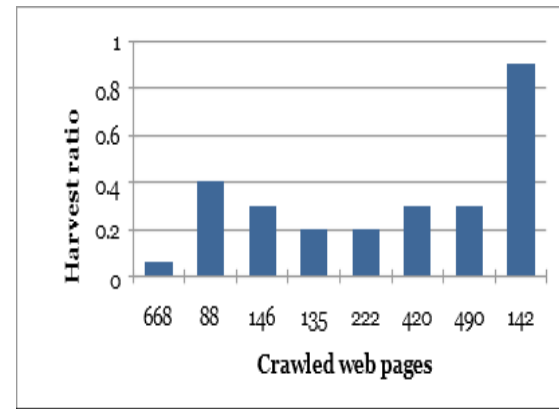
(a): Number Of Pages For Business Category



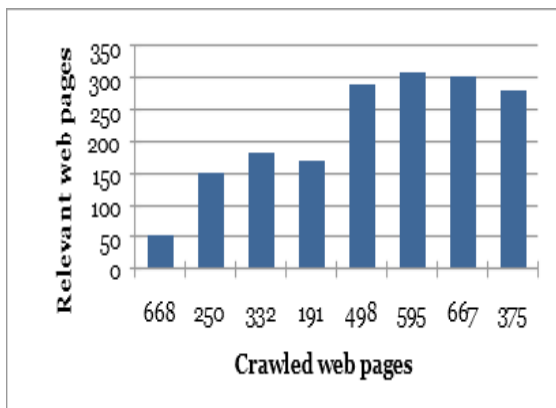
(b): Harvest Ratio For Business Category



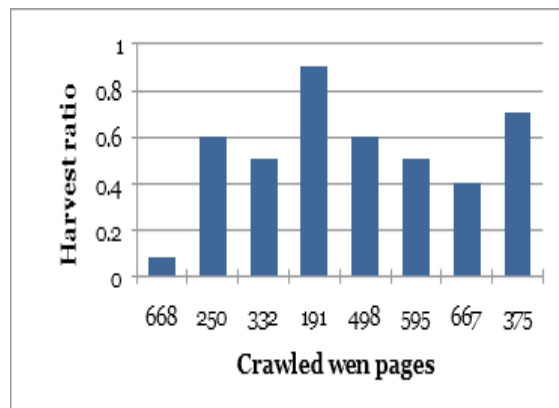
(c): Number Of Pages For Crime Category



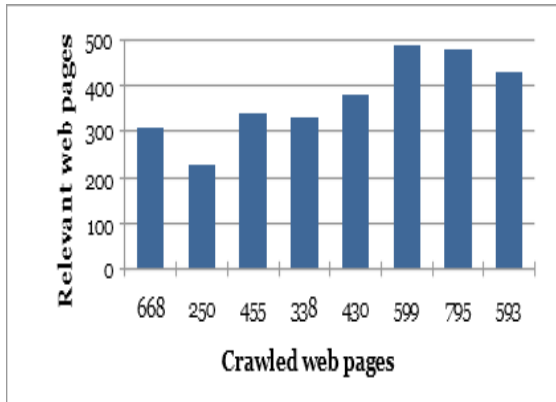
(d): Harvest Ratio For Crime Category



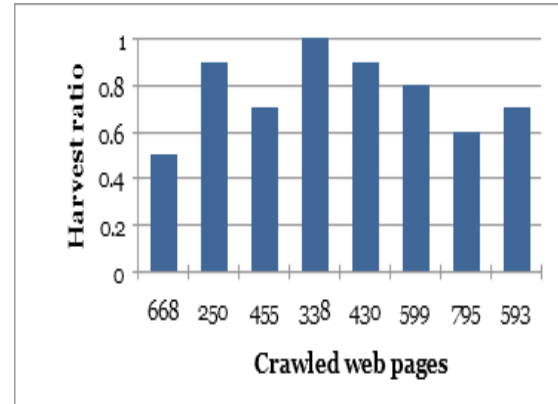
(e): Number Of Pages For Health Category



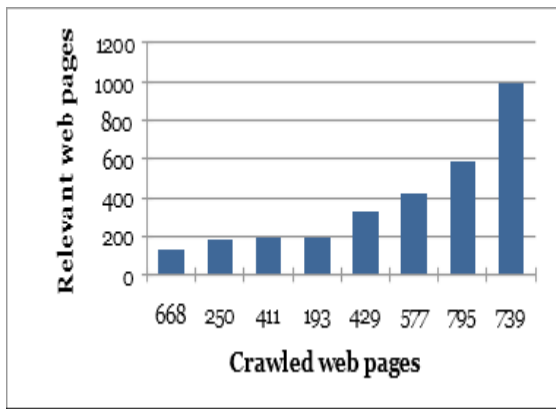
(f): Harvest Ratio For Health Category



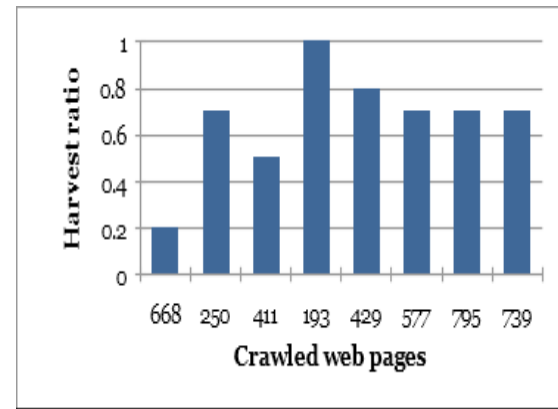
(g): Number Of Pages For Politics Category



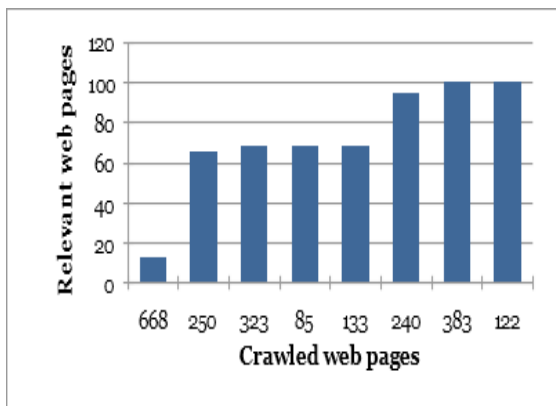
(h): Harvest Ratio For Politics Category



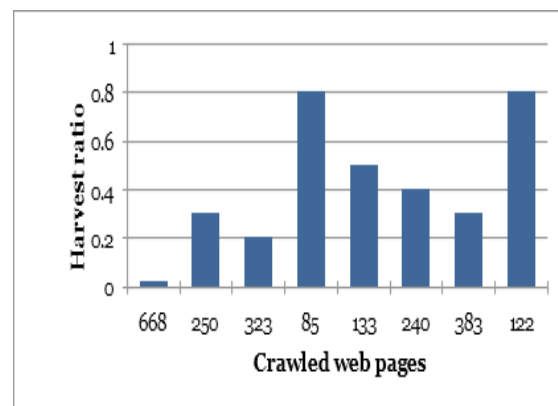
(i): Number Of Pages For Sports Category



(j): Harvest Ratio For Sports Category



(k): Number Of Pages For Technology Category



(l): Harvest Ratio For Technology Category

Figure 5.1: Number of Releive Web Pages and Harvest Ratio For All News Categories

Based on the results from the harvest ratio which shown in Figure 5.1, we observe that:

1. In all categories, the harvest ratio at the first phase was limited. To explain, at this phase, 668 web pages were crawled and distributed between all categories while during the other seven stages, each category will be crawled individually.
2. Harvest ratio for the four categories of politics, sports, business, and health is considered very

high compared to the other two categories of crime and technology. As mentioned previously, this is due to the density of the links and their distribution inside the seed websites that have been chosen.

3. The non-stability of harvest ratio values as it varies with the number of pages crawled. To solve this problem and increase the harvest ratio, (Choudhary and Roy, 2013) proposed a semantic focused web crawler and compared its performance to the basic crawler, focused crawler, and priority-based focused crawler using harvest ratio. The performance of the three crawlers was unstable too, while the semantic crawler outperformed them, even when its performance decreased, it remained more stable. This work has been mentioned in chapter 2 section 2.4.

5.3.3 Changes of News Categories Web Pages:

In this section, we will display the total number of pages whose category has been changed as all pages re-categorised before their changes are detected and then displayed the total changes that have been made to the web pages during crawling phases.

5.3.3.1 Re-Categorising Categories Web Pages

As has been discussed in chapter 4 section 4.2.8, before recrawling web pages, the process of re-categorising the previously stored web pages will be performed. Figure 5.2 exhibits for each category the percentage of pages that their categories have changed all over the seven crawling stages, whereas for first stage, there are still no pages. The highest rate was for politics category while the lowest was for technology category for its web pages were steady and have not changed.

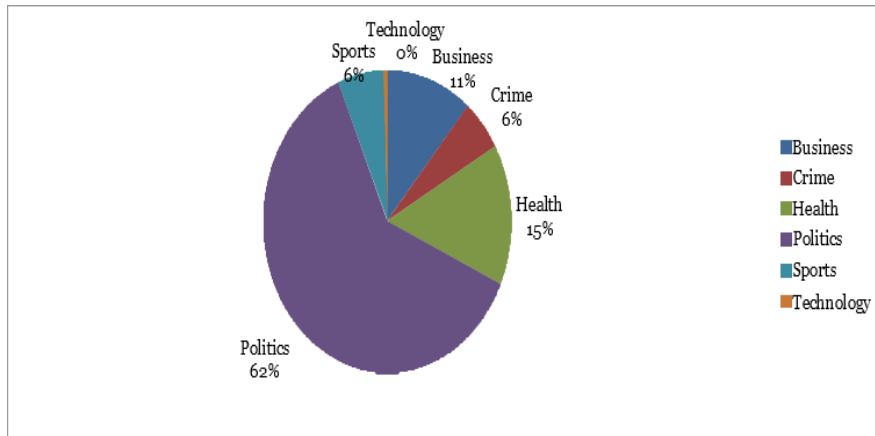


Figure 5.2: Percentage of Web Pages Their Category Has Changed Through the seventh Crawling Phases

All the pages had their categories changed, either because of the dynamicity of their text or

because of changes in the terms representing the category between the crawling phases (as discussed in chapter 4 section 4.2.3). As a result of re-categorising web pages, the number of URLs for some categories will sometimes be reduced but URLs of other categories will not necessarily be increased, and that is because web pages similarity values are less than the category threshold. Hence these re-categorised web pages will be eliminated and not attached to the category that these pages changed to.

For example, referring to Figure 5.1.c, we note that the number of pages in phase 4 has been reduced from 38 to 33. The reason is that five pages had their category changed to political category, three have been deleted and two accepted and added to the politics category.

5.3.3.2 Structural and Textual Changes of Categories Web pages

The web crawler revisits web pages to recognise whether they had modified structurally or textually. These modifications will be used as the importance metrics, together with the similarity values of web pages to prioritise their URLs inside the frontier. Figure 5.3 shows the percentages of total structural and textual modifications appearing in all categories web pages throughout all the crawling phases.

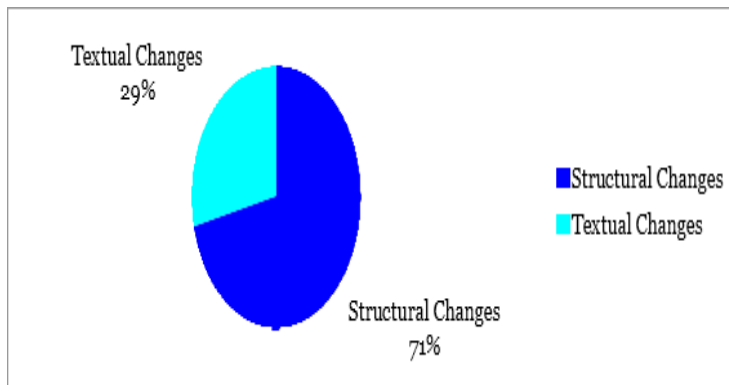
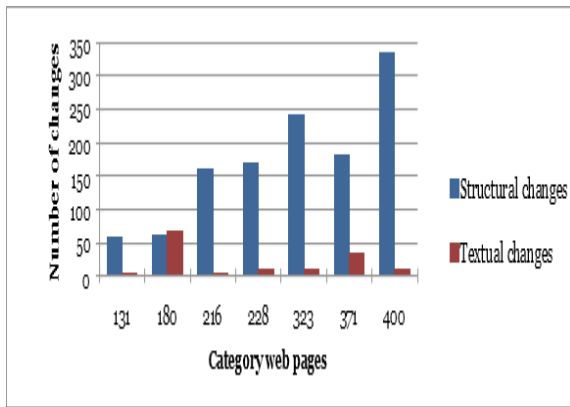


Figure 5.3: Total of Structural and Textual Changes Through Crawling Phases

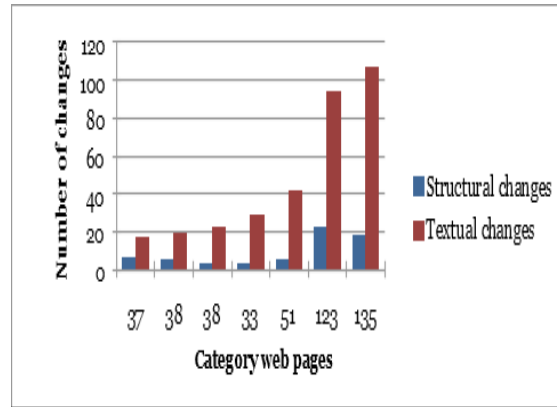
It is obvious that the structural changes exceed the textual changes. Therefore, upon knowing the method used in this search, any web page has structurally changed can be textually changed as the text-changing page is structurally non-volatile. Besides, consideration was being given to whether there had been any structural changes in advance before considering the text.

Figure 5.4 shows the total structural and textual changes for all categories through the seven crawling phases, where the horizontal axis denotes the number of relevant pages entered from the

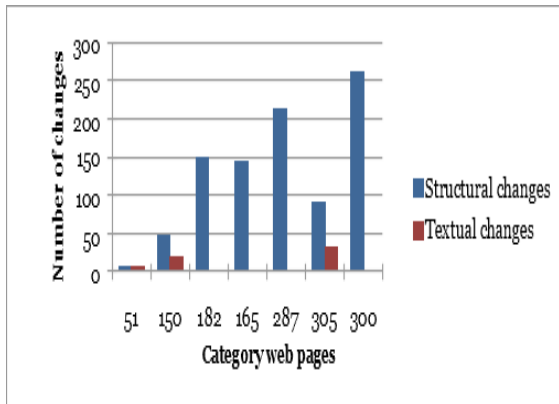
previous phase and the vertical axis denotes the number of structural and textual changes that have appeared in the crawled pages. For example, in Figure 5.4.a at the second phase of the business category, the crawler receives 131 URLs collected from the first phase, and as we can observe 63 (46%) out of 131 web pages have changed at the second phase. Following the finished crawling of the second phase, the obtained 180 web pages will be used as input for the third phase and so on.



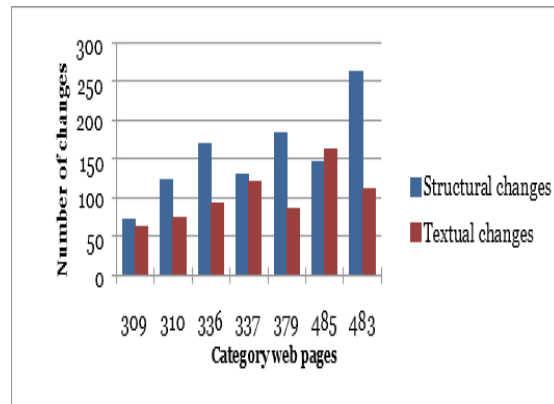
(a): Changes for Business Category Pages



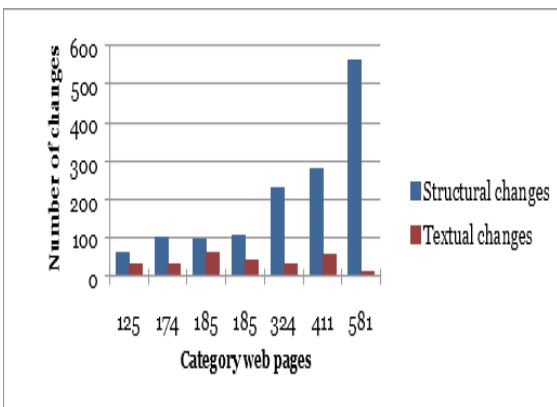
(b): Changes for Crime Category Pages



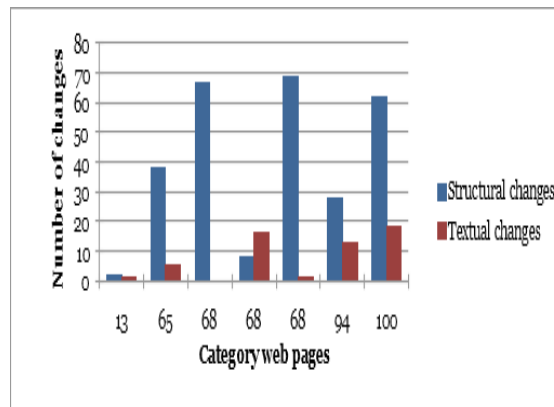
(c): Changes for Health Category Pages



(d): Changes for Politics Category Pages



(e): Changes for Sports Category Pages



(f): Changes for Technology Category Pages

Figure 5.4: Total Changes Appeared in Categories Web Pages

As appearing in the previous Figures, all the categories were structurally and textually changed, with the most of changes being structural, except for the category of crime where the structural changes were almost completely non-existent, with most of the changes being textual; in addition to the political category where all the changes had almost high rates through all stages.

5.4 The Frontier

After listing the results of the web pages classification process and counting the number of times web pages had changed. Both the similarity and changes of web pages used as importance metrics to prioritise the URLs within the frontier during crawling phases.

During each crawling phase, two static and dynamic frontiers are configured where the URLs in the dynamic frontier are of the highest importance so they will be crawled first followed by the static frontier URLs. Figure 5.5 shows the total dynamic and static URLs for each category at the beginning of each crawling phase. From the Figure, we notice that the more we crawl, the greater the number of dynamic URLs we get.

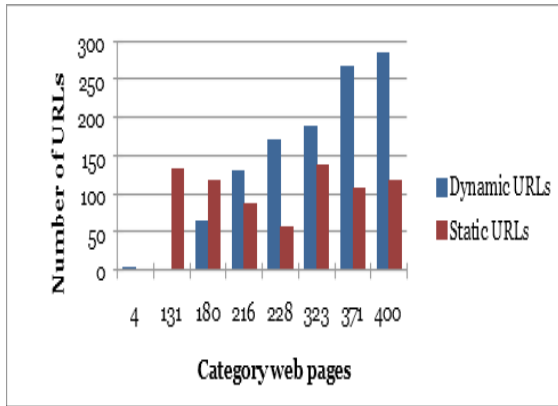


Fig 5.1: Dynamic and Static URLs for Business Category

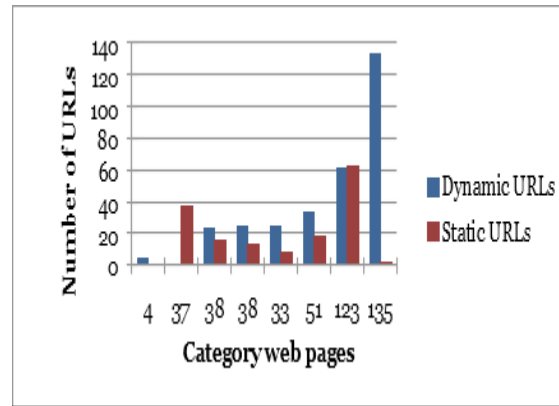


Fig 5.2: Dynamic and Static URLs for Crime Category

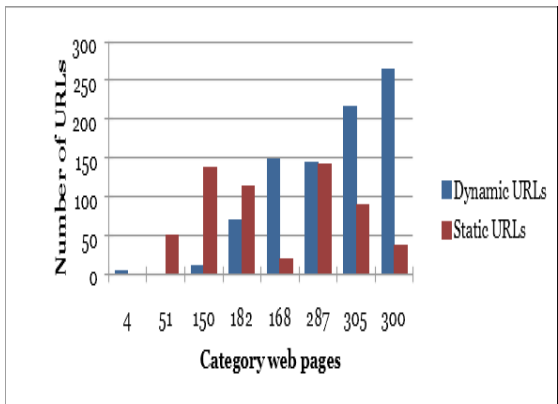


Fig 5.3: Dynamic and Static URLs for Health Category

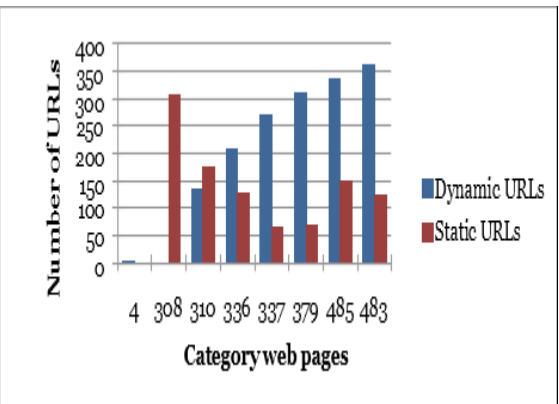


Fig 5.4: Dynamic and Static URLs for Politics Category

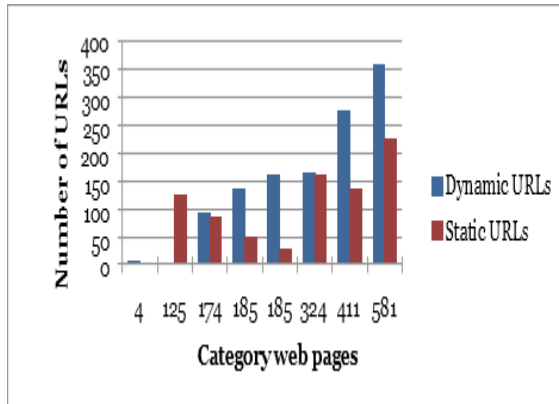


Fig 5.5: Dynamic and Static URLs for Sports Category

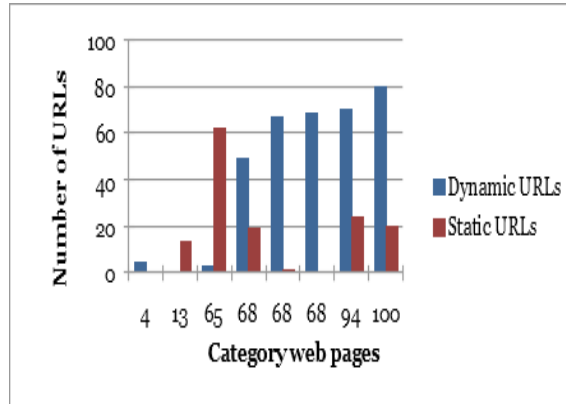


Fig 5.6: Dynamic and Static URLs for Technology Category

Figure 5.5: Dynamic and Static URLs For All Categories Through All Crawling Phases

During all crawling phases, a queue named priority queue is created where each one of the crawled URLs that extracted from the frontier is appended together with its priority value to this queue based on its crawling order. When the execution is terminated, this queue is saved inside a file, and due to a large number of URLs, Figure 5.6 displays a portion of screenshots of these files during the first crawling phase, as well as the middle and final phases of the sports category.

	A	B
1	URL	PRIORITY VALUE
2	http://www.bbc.com/	1
3	https://www.aljazeera.com/	1
4	https://www.theguardian.com/international	1
5	https://abcnews.go.com/	1
6	http://www.theguardian.com/us-news/2018/nov/04/republican-race-baiting-midterm-elections-donald-trump	0.27154
7	http://www.theguardian.com/world	0.254
8	http://abcnews.go.com/WN	0.20739
9	http://www.theguardian.com/us-news/2018/nov/04/illinois-republican-peter-roskam-sean-casten-trump	0.20739
10	http://www.theguardian.com/commentisfree/2018/nov/04/the-american-civil-war-didnt-end-and-trump-is-a-confederate-president	0.1905
11	http://www.theguardian.com/us-news/2018/nov/04/miami-cubans-midterm-elections-voters	0.1905
12	http://www.aljazeera.com/news	0.17961
13	http://www.bbc.com/news/10462520	0.17961
14	http://www.bbc.com/news/world_radio_and_tv	0.22361
15	http://www.bbc.com/news/business-38507481	0.22361
16	http://www.bbc.com/news/business-45263511	0.22361
17	http://www.bbc.com/news/topics/cg5rv39y9mmt/global-trade	0.22361
18	http://www.bbc.com/news/business-45255623	0.19365
19	http://www.bbc.com/news/business-44711257	0.18257
20	http://www.aljazeera.com/investigations	0.17961
21	http://www.aljazeera.com/investigations/reporters.html	0.22361
22	http://www.aljazeera.com/topics/events/midterms-2018.html	0.17961

(a): Portion of URLs in Priority Queue at The First Crawling Phase

A		B
1	URL	PRIORITY VALUE
2	http://www.bbc.com/sport/cricket/live-scores	3.24031
3	http://www.bbc.com/sport/rugby-league	3.23305
4	http://www.bbc.com/sport/football/45823981	3.22791
5	http://www.bbc.com/sport/rugby-union/teams/sale	3.22707
6	http://www.bbc.com/sport/swimming/45503795	3.22497
7	http://www.bbc.com/sport/cycling/45973736	3.22229
8	http://www.bbc.com/sport/rugby-league/teams/london-broncos	3.22186
9	http://www.bbc.com/sport/hockey	3.22167
10	http://www.bbc.com/sport/basketball	3.21803
11	http://www.bbc.com/sport/rugby-league/14507071	3.21573
12	http://www.bbc.com/sport/olympics	3.21518
13	http://www.bbc.com/sport/football/scottish-challenge-cup	3.21316
14	http://www.bbc.com/sport/tennis	3.21257
15	http://www.bbc.com/sport/cricket/34279619	3.21229
16	http://www.bbc.com/sport/football/45852287	3.21224
17	http://www.bbc.com/sport/swimming	3.21213
18	http://www.bbc.com/sport/football/scottish-league-cup	3.21146
19	http://www.bbc.com/sport/rugby-union/46030812	3.21146
20	http://www.bbc.com/sport/cricket/14932847	3.21091
21	http://www.bbc.com/sport/football/fa-cup	3.21084
22	http://www.bbc.com/sport/cricket/19649043	3.21004
23	http://www.bbc.com/sport/cricket/34172677	3.20816
24	http://www.bbc.com/sport/baseball	3.20651

(b): Portion of Dynamic URLs in Priority Queue at The Fourth Crawling Phase for Sports News Category

A		B
1	URL	PRIORITY VALUE
2	http://www.bbc.com/sport/rugby-union/live-scores	0.24941
3	http://www.bbc.com/sport/rugby-union/european-cup/live-scores	0.24941
4	http://www.bbc.com/sport/rugby-union/european-challenge-cup/live-scores	0.24941
5	http://www.bbc.com/sport/rugby-union/rugby-championship/live-scores	0.24941
6	http://www.bbc.com/sport/live/football/45682889	0.24582
7	http://www.bbc.com/sport/rugby-league/live-scores	0.2443
8	http://www.bbc.com/sport/rugby-league/challenge-cup/live-scores	0.2443
9	http://www.bbc.com/sport/rugby-league/league-one-cup/live-scores	0.24384
10	http://www.bbc.com/sport/rugby-union/premiership-rugby-cup/live-scores	0.23124
11	http://www.bbc.com/sport/football/leagues-cups	0.22659
12	http://www.bbc.com/sport/football/46130548	0.22383
13	http://www.bbc.com/sport/rugby-league/teams	0.21683
14	http://www.bbc.com/sport/rugby-union/teams	0.21572
15	http://www.bbc.com/sport/all-sports	0.21487
16	http://www.bbc.com/sport/rugby-league/league-one/live-scores	0.21483
17	http://www.bbc.com/sport/cricket/teams	0.21463
18	http://www.bbc.com/sport/football/46140699	0.19803
19	http://www.bbc.com/sport/get-inspired/23179331	0.18934
20	http://www.bbc.com/sport/rugby-league/46193469	0.18787
21	http://www.bbc.com/sport/av/rugby-league/45310995	0.19803
22	http://www.bbc.com/sport/rugby-league/teams/catalans	0.19803
23	http://www.bbc.com/sport/rugby-league/45310995	0.19803
24	http://www.aljazeera.com/topics/subjects/basketball.html	0.1826
25	http://abcnews.go.com/topics/sports/nfl.htm	0.18221
26	http://www.aljazeera.com/topics/subjects/us-sports.html	0.17831

(c): Portion of Static URLs in Priority Queue at The Fourth Crawling Phase for Sports News Category

A		B
1	URL	PRIORITY VALUE
2	http://www.bbc.com/sport/cricket/live-scores	7.24031
3	http://www.bbc.com/sport/rugby-league	7.23305
4	http://www.bbc.com/sport/football/45823981	7.22791
5	http://www.bbc.com/sport/rugby-union/teams/sale	7.22707
6	http://www.bbc.com/sport/swimming/45503795	7.22497
7	http://www.bbc.com/sport/cycling/45973736	7.22229
8	http://www.bbc.com/sport/hockey	7.22167
9	http://www.bbc.com/sport/basketball	7.21803
10	http://www.bbc.com/sport/rugby-league/14507071	7.21573
11	http://www.bbc.com/sport/olympics	7.21518
12	http://www.bbc.com/sport/football/scottish-challenge-cup	7.21316
13	http://www.bbc.com/sport/tennis	7.21257
14	http://www.bbc.com/sport/cricket/34279619	7.21229
15	http://www.bbc.com/sport/football/45852287	6.21224
16	http://www.bbc.com/sport/swimming	6.21213
17	http://www.bbc.com/sport/football/scottish-league-cup	6.21146
18	http://www.bbc.com/sport/cricket/14932847	6.21091
19	http://www.bbc.com/sport/football/fa-cup	6.21084
20	http://www.bbc.com/sport/cricket/19649043	6.21004
21	http://www.bbc.com/sport/cricket/34172677	6.20816
22	http://www.bbc.com/sport/baseball	6.20651
23	http://www.bbc.com/sport/football/champions-league	6.20376
24	http://www.bbc.com/sport/football/29458784	6.20364
25	http://www.bbc.com/sport/rugby-union	6.20313

(d): Portion of Dynamic URLs in Priority Queue at The Eighth Crawling Phase for Sports News Category

	A	B
1	URL	PRIORITY QUEUE
2	http://www.bbc.com/sport/american-football/46272417	0.27228
3	http://www.theguardian.com/sport/2018/dec/19/missy-franklin-swimming-retirement-olympics	0.26463
4	http://www.bbc.com/sport/rugby-union/rugby-championship/fixtures	0.25085
5	http://www.bbc.com/sport/rugby-union/european-cup/live-scores	0.24941
6	http://www.bbc.com/sport/rugby-union/european-challenge-cup/live-scores	0.24941
7	http://www.bbc.com/sport/rugby-union/rugby-championship/live-scores	0.24941
8	http://www.bbc.com/sport/15561688	0.24819
9	http://www.bbc.com/sport/football/46264663	0.24637
10	http://www.bbc.com/sport/football/46249985	0.24474
11	http://www.bbc.com/sport/cricket/43914118	0.24439
12	http://www.bbc.com/sport/rugby-league/live-scores	0.2443
13	http://www.bbc.com/sport/rugby-league/challenge-cup/live-scores	0.2443
14	http://www.bbc.com/sport/rugby-league/league-one-cup/live-scores	0.24384
15	http://www.bbc.com/sport/rugby-union/22381109	0.24191
16	http://www.bbc.com/sport/football/45639200	0.24174
17	http://www.bbc.com/sport/football/46242693	0.24122
18	http://www.bbc.com/sport/rugby-union/european-cup/fixtures	0.23992
19	http://www.bbc.com/sport/rugby-union/match/ERUP4039122	0.23637
20	http://www.bbc.com/sport/rugby-union/match/ERUP4039118	0.2358
21	http://www.bbc.com/sport/rugby-union/match/ERUP4039120	0.23525
22	http://www.bbc.com/sport/rugby-league/13629851	0.23432
23	http://www.bbc.com/sport/football/40004559	0.23288
24	http://www.bbc.com/sport/rugby-union/46222074	0.23247
25	http://www.bbc.com/sport/hockey/46252082	0.2317
26	http://www.bbc.com/sport/rugby-union/premiership-rugby-cup/live-scores	0.23124

(e): Portion of Static URLs in Priority Queue at The Eighth Crawling Phase for Sports News Category

Figure 5.6: Portions of Dynamic and Static URLs at First, Middle and Final Crawling Phases

Based on the results observed from the priority queue, it is clear that all dynamic URLs crawled first, followed by static URLs based on their priority values.

As for the dynamic frontier, there are web pages, which considered not dynamic while their dynamicity is high as in (<http://www.aljazeera.com/news/2017/01/donald-trump-russian-hacking-election-outcome-170107034647205.html>) web page, in the case of these pages there are parts of them, that are changing continuously making its dynamicity higher while the rest of the page is static. Figure 5.7 shows the most dynamic parts of the page which are trending and tweeting parts.

The image shows a screenshot of a news article from Al Jazeera. The article text includes: "Trump met the heads of the Office of the Director of National Intelligence, the Central Intelligence Agency, the Federal Bureau of Investigation and the National Security Agency in New York on Friday, on their newly completed report into Moscow's alleged interference." Below this is a tweet from Donald J. Trump: "Gross negligence by the Democratic National Committee allowed hacking to take place. The Republican National Committee had strong defenses!" To the right of the tweet, an arrow points to the text "Web page dynamic parts". Below the tweet is a "TRENDING" sidebar with four items: "US 'a highly unreliable' partner to Ankara: Turkish official", "Lab strips DNA pioneer James Watson of honours for racist views", "In the US South, anti-Confederate protesters face harassment", and "'China is after us': Uighurs in Pakistan report intimidation".

Figure 5.7: Dynamic Parts of (<http://www.aljazeera.com/news/2017/01/donald-trump-russian-hacking-election-outcome-170107034647205.html>) Web Page

5.5 Dynamicity approach versus (*PageRank, Backlink Count and Forward Link Count*)

As discussed in chapter 3 section 3.4 that on the contrary, other methods used by crawlers like: PageRank, Backlink Count and Forward Link Count which give priority to the page based on its importance to the other pages unlike the proposed method in this dissertation, which examines the dynamicity of the page itself without its relation to other pages. Tables (5.2) to (5.7) shows the five top most dynamic pages in each category and their priority order in PageRank, Backlink and Forward algorithms.

Table 5.2: Priority Order of Top Five Most Dynamic Page in Business Category Compared with PageRank, Backlink Count and Forward link Count Algorithms

URL	Dynamicity	Pagerank	Backlinks	Forward
http://www.bbc.com/news/business-46053283	1	86	98	312
http://www.theguardian.com/business/global-economy	2	87	193	265
http://www.bbc.com/news/business-36544970	3	88	100	372
http://www.theguardian.com/business/stock-markets	4	89	182	195
http://www.bbc.com/news/business-46068270	5	90	276	313

Table 5.3: Priority Order of Top Five Most Dynamic Page in Crime Category Compared with PageRank, Backlink Count and Forward link Count Algorithms

URL	Dynamicity	Pagerank	Backlinks	Forward
http://www.aljazeera.com/news/2018/10/germany-arrests-suspect-killing-viktoria-marinova-181010093524925.html	1	38	126	20
http://abcnews.go.com/US/wireStory/convict-convicted-fatal-shooting-tennessee-officer-58961664	2	100	45	114
http://www.aljazeera.com/news/2018/10/suspect-arrested-bulgarian-journalist-murder-report-181009113724699.html	3	86	180	32
http://www.aljazeera.com/news/2018/03/bolivia-prisoners-killed-police-raid-notorious-jail-180315080551666.html	4	184	43	103
http://www.aljazeera.com/news/2018/09/philippine-mayor-duterte-drug-list-shot-dead-gunmen-180905002148688.html	5	93	181	62

Table 5.4: Priority Order of Top Five Most Dynamic Page in Health Category Compared with PageRank, Backlink Count and Forward link Count Algorithms

URL	Dynamicity	Pagerank	Backlinks	Forward
http://abcnews.go.com/Health	1	10	2	109
http://www.theguardian.com/science/cancer	2	20	58	15
http://www.theguardian.com/science/medical-research	3	21	57	75
http://www.theguardian.com/society/cancer	4	22	90	23
http://www.theguardian.com/science/drugs	5	23	103	40

Table 5.5: Priority Order of Top Five Most Dynamic Page in Political Category Compared with PageRank, Backlink Count and Forward link Count Algorithms

URL	Dynamicity	Pagerank	Backlinks	Forward
http://www.aljazeera.com/news/2017/01/donald-trump-russian-hacking-election-outcome-170107034647205.html	1	261	335	189
http://www.aljazeera.com/news/2018/09/mid-term-elections-key-issues-180928103258276.html	2	11	42	421
http://abcnews.go.com/Politics	3	28	3	371
http://abcnews.go.com/WNT/video/president-obama-gave-heated-speech-trump-ahead-midterm-57712195	4	292	349	406
http://www.aljazeera.com/news/2016/07/obama-clinton-qualified-president-160727233030311.html	5	293	225	182

Table 5.6: Priority Order of Top Five Most Dynamic Page in Sports Category Compared with PageRank, Backlink Count and Forward link Count Algorithms

URL	Dynamicity	Pagerank	Backlinks	Forward
http://www.bbc.com/sport/cricket/live-scores	1	43	27	513
http://www.bbc.com/sport/rugby-league	2	10	7	83
http://www.bbc.com/sport/football/45823981	3	603	438	412
http://www.bbc.com/sport/rugby-union/teams/sale	4	238	493	122
http://www.bbc.com/sport/swimming/45503795	5	239	466	366

Table 5.7: Priority Order of Top Five Most Dynamic Page in Technology Category Compared with PageRank, Backlink Count and Forward link Count Algorithms

URL	Dynamicity	Pagerank	Backlinks	Forward
http://www.theguardian.com/technology/elon-musk	1	67	91	73

URL	Dynamicity	Pagerank	Backlinks	Forward
http://www.theguardian.com/technology/data-protection	2	81	96	6
http://abcnews.go.com/Technology	3	3	2	90
http://www.theguardian.com/uk/technology	4	9	83	99
http://www.theguardian.com/technology/samsung	5	4	40	78

After implementing and comparing the algorithms mentioned above, the results indicate that:

1. The most dynamic web pages sometimes have the lowest priority in PageRank, Backlink Count and Forward Link Count algorithms.
2. As for the PageRank algorithm, the more we progress in the crawling phases, the greater the number of pages that refer to each other we get, which means increasing their importance and thus increase their priority. For example (<http://abcnews.go.com/Health>) web page when the PageRank algorithm implemented after the first crawling phase, its priority was 147, while after the eighth phases the priority as shown in Table 5.4, which was 10.

As for the time performance between the PageRank and our proposed method, the time difference was very wide, where at the eighth phase of the crawling, PageRank algorithm required 35 hours while the proposed method required 341 milliseconds to calculate web pages' scores and reorder them. This is because of the computational complexity of the PageRank algorithm to determine the importance of web pages while in the proposed method is just required to collect web pages similarity with their dynamicity.

5.6 Summary

The terms representing news categories, the total number of URLs for each news category and number of changes appeared in their web pages was displayed in this chapter. Results of the frontier were displayed and shown that the proposed method gives promised results in crawling the most dynamic web pages first and finally a brief comparison between the proposed metric and other metrics used by the crawler is performed and shown that even if the page having the highest dynamicity in some cases will crawl later not earlier in the other methods.

Chapter Six

Conclusion and Future Works

We conclude in this chapter the dissertation and suggest some future works.

6.1 Conclusion

This dissertation is interested in the problem of crawlers, namely how a crawler gives priority to web pages so that the most important ones are crawled before the least important. The crawler revisits pages to discover changes and to provide search engines with fresh copies. In fact, due to the problems facing web crawlers, they should try to explore the changes that have occurred on the most important pages first before the less important ones.

In this dissertation, a new strategy to prioritise web pages in web crawling is proposed. Therefore, we designed a focused web crawler that utilises similarity and dynamicity of web pages in assigning importance to them. On that account, the structural and textual changes of the pages are enumerated and the web pages are crawled from top to bottom. However, when concerning the textual changes in the web pages instead of looking at the whole text of the web page, we proposed that the crawler will look at the text of the most unsteady parts of the web page and will neglect the rest. In addition, two frontiers are configured. Dynamic frontier for dynamic URLs and static frontier for static URLs. And since the dynamic pages have the highest priority, the URLs of the dynamic frontier are crawled first and then followed by static URLs. The results from the crawler implementation were effective in showing that the higher dynamic pages took precedence in crawling.

6.2 Future Works

The Future work includes the following:

1. **Crawl JavaScript Pages:** currently most sites are built using JavaScript language. As a result, a large part of the script tags are embedded within web pages. The script tags contain codes to add and edit page content, either structurally or textually. In this dissertation we have created HTML crawler which extracts only the HTML code of the web pages and neglects script tags, for example, CNN website are built using the JavaScript, when the crawler crawls its web pages the crawler will not get any codes from the pages except their main parts such as the menu and the footer. Additionally, in future research, it would be interesting to create a web crawler, which

can recognise and read these kind of web pages and detect the most dynamic (changing) part of them either structurally or textually instead of reading the entire page.

2. **Detecting Presentation Changes:** detecting presentation changes, which occur when the appearance of the page changes.
3. **Parallel Web Crawler:** since we use a set of news categories and crawl each category individually. Therefore, to speed up the crawling, the parallel crawling process needs to be implemented to crawl all categories at once to reduce the crawling time.

References

- Adar, E., Teevan, J., Dumais, S. and Elsas, J. (2009). The web changes everything: understanding the dynamics of web content. In: Proceedings of the Second ACM International Conference on Web Search and Data Mining, (pp. 282-291). February 09 - 12. Barcelona, Spain: ACM Publishing.
- Assis, G., Laender, A. and Gonçalves, M. (2008). The impact of term selection in genre-aware focused crawling. In: Proceedings of the 2008 ACM Symposium on Applied Computing, (pp. 1158-1163). March 16 - 20. Fortaleza, Ceara, Brazil: ACM Publishing.
- Baker, M., & Akcayol, M.(2017). Priority queue based estimation of importance of web pages for web crawlers. *International Journal of Electrical and Computer Engineering*, 9(1), 330-342.
- Batsakis, S., Petrakis, E. and Milios, E. (2009). Improving the performance of focused web crawlers. *Journal of Data & Knowledge Engineering*, 68(10), 1001-1013.
- Bhatt, D., Vyas, D. and Pandya, S. (2015). Focused web crawler. *Journal of Advances in Computer Science and Information Technology*, 2(11), 1-6.
- Bracewell, D., Yan, J., Ren, F. and Kuroiwa S. (2009). Category classification and topic discovery of Japanese and english news articles. *Journal of Electronic Notes in Theoretical Computer Science*, 255, 51-65.
- Capella University. (2007). *Grammar Handbook*. Minneapolis, Minnesota: Author
- Cho, J., Garcia-Molina, H. and Page, L. (1998). Efficient crawling through URL ordering. In: Proceedings of the seventh international conference on World Wide (pp. 161-172). Brisbane, Australia: ACM Publishing.

- Choi, B., and Yao, Z. (2005). Web page classification. In: Chu. W and Lin. T (Eds.), Foundations and Advances in Data Mining. Studies in Fuzziness and Soft Computing (pp. 221-274). Berlin, Heidelberg: Springer.
- Choudhary, J., & Roy, D. (2013). Priority based semantic web crawler. International Journal of Computer Applications, 81(15), 0975–8887.
- Dutta, M., & Bansal, K.(2016). A review paper on various search engines (Google, Yahoo, Altavista, Ask and Bing). International Journal on Recent and Innovation Trends in Computing and Communication, 4(8), 190-195.
- Goel, S., & Aggarwal, R.(2012). An efficient algorithm for web page change detection. International Journal of Computer Applications, 48(10), 0975-888.
- Kumar, R., & Jain, A. (2014). Efficient crawling through dynamic priority of web pages in site-map. Informatics Engineering, an International Journal (IEIJ), 2(2), 1-11.
- Lee, D., Chuang, H., & Seamons, K. (1997). Document ranking and the Vector Space Model. Journal of IEEE Software, 14(2), 67-75.
- Manning, C., Raghavan, P. and Schütze, H. (2008). Scoring, term weighting and the vector space model. Introduction to information retrieval (pp. 109-133). Cambridge: Cambridge University Press.
- Porter. M., (1980). An algorithm for suffix stripping. Program, 14(3) .130-137.
- Singhal, V., & Sharma, S. (2012) Text content based web page refresh policy. Journal of Global Research in Computer Science, 3(11), 32-37.

Siqueira, G., Assis, G., Ferreira, A., Silva, A., Mangaravite, V and Pádua, F. (2017). Strategies for automatic determination of similarity threshold for Genre-Aware focused crawling processes. IADIS International Journal on WWW/Internet, 15(1), 15-30.

Udapure, T., Kale, R. & Dharmik, R. (2014). Study of web crawler and its different types. Journal of Computer Engineering, 16(1), 01-05.

Wood, L., Hors, A., Apparao, V., Byrne, S., Champion, M., Isaacs, S., Jacobs, I., Nicol, G., Robie, J., Sutor, R. and Wilson, C. (Eds.). (1998). Document Object Model (DOM) Level 1 Specification, Version 1.0. World Wide Web Consortium Recommendation.

Appendix A

The Overall Crawling Process of Classic Focused Crawler

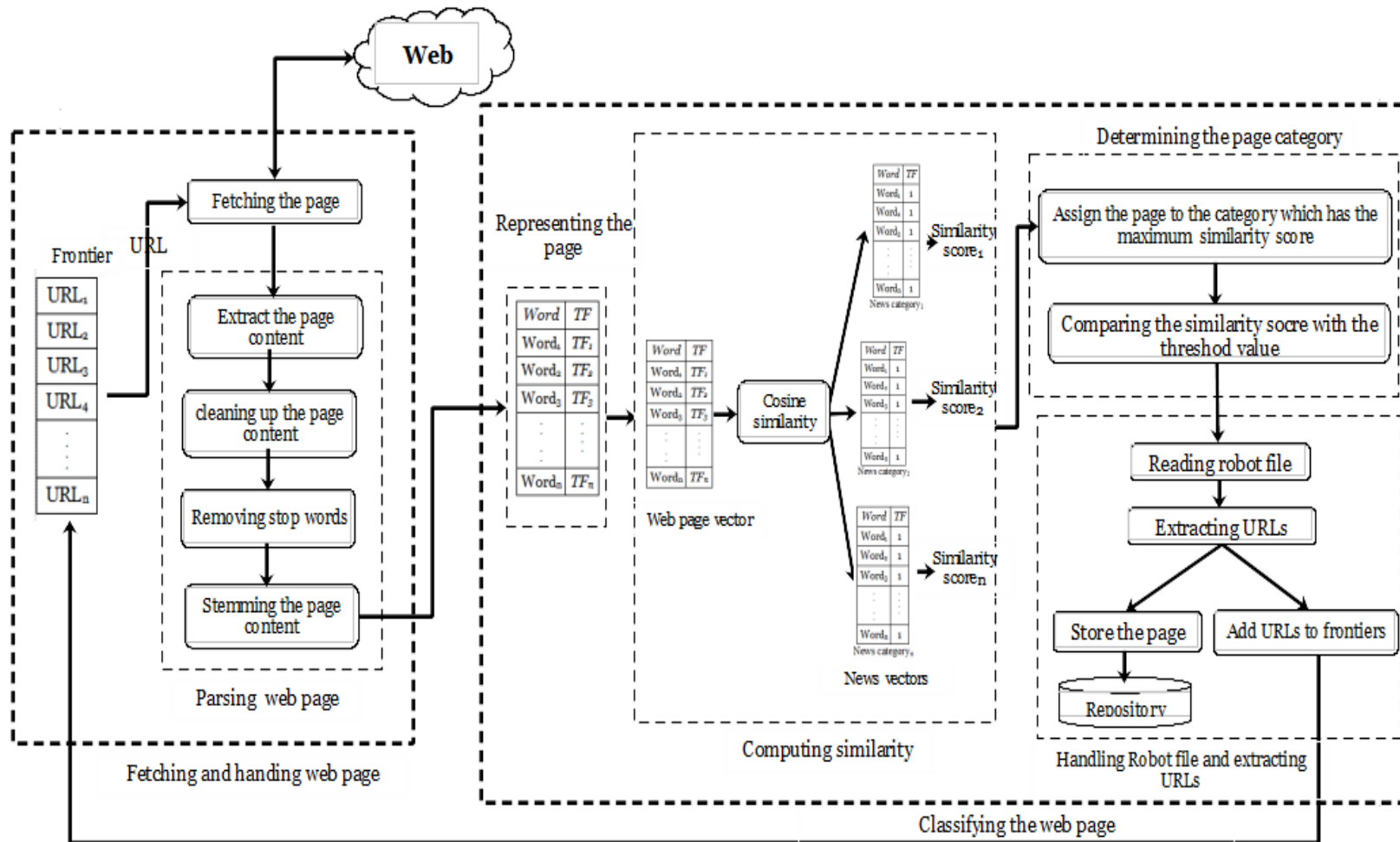
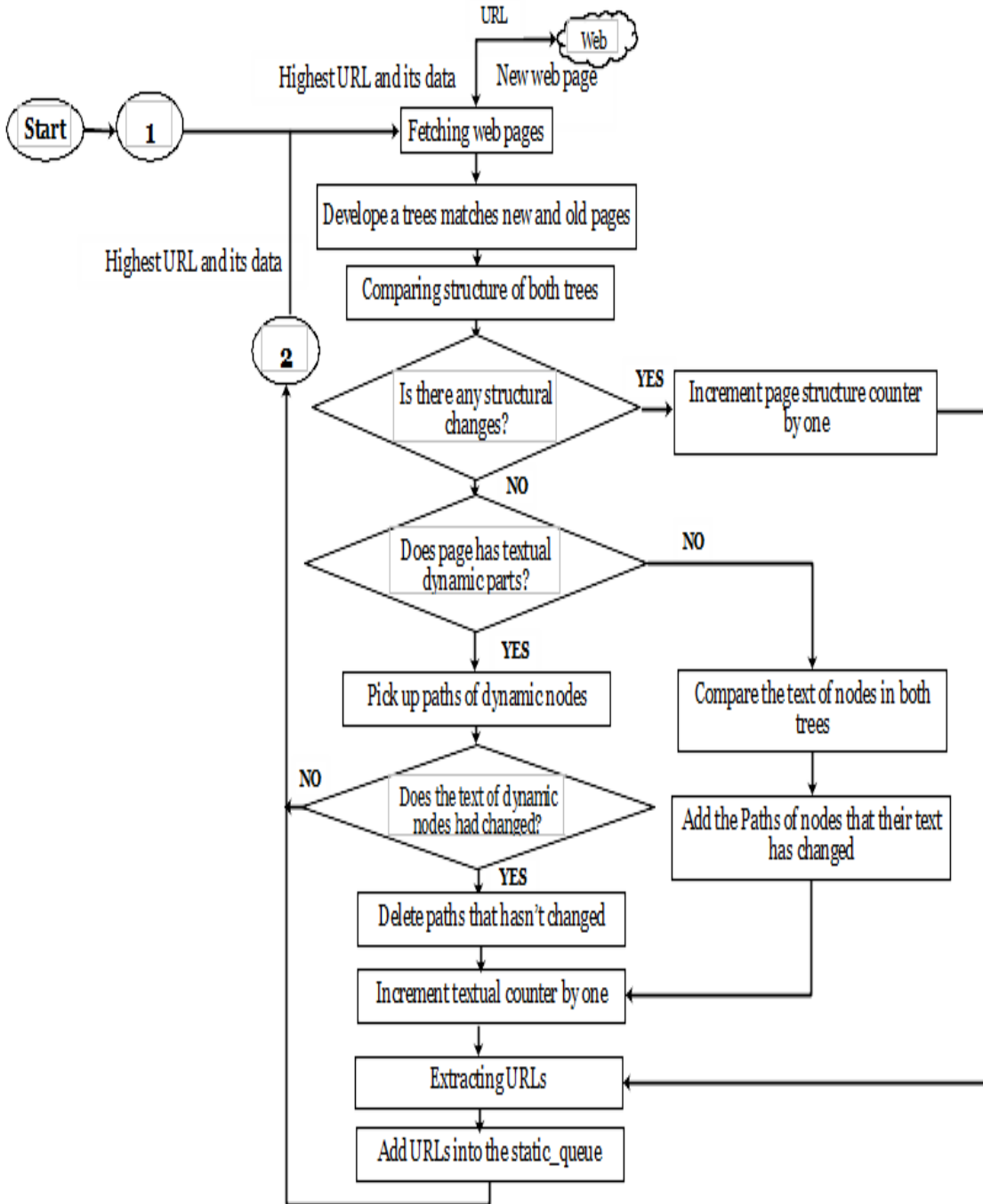


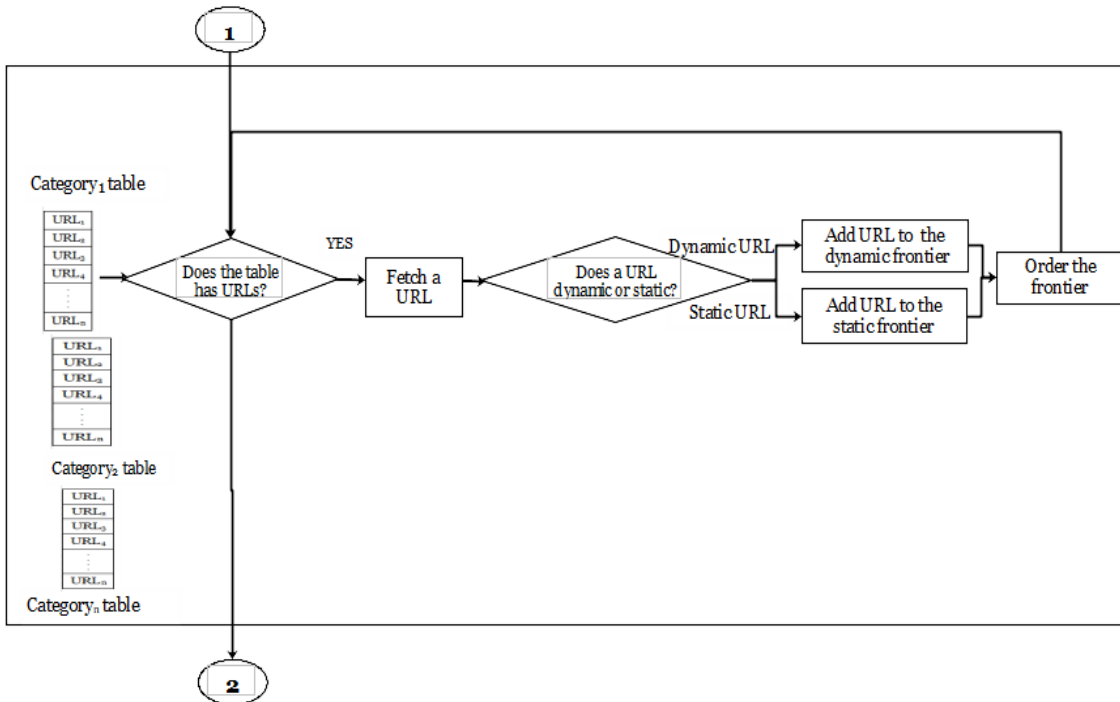
Figure A.1: The Overall Crawling Process of Classic Focused Crawler

Appendix B

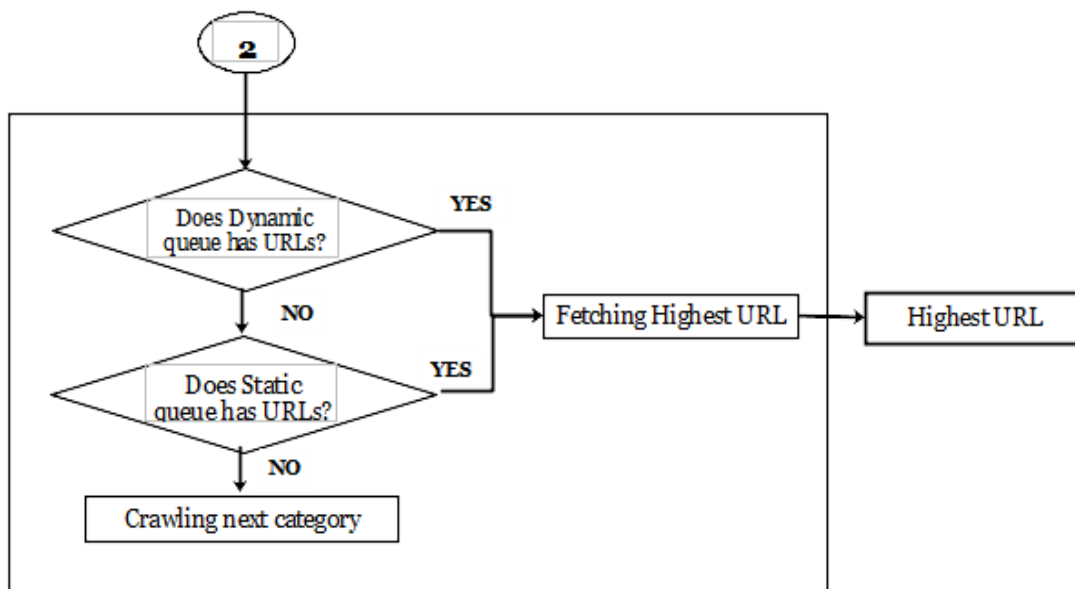
The Overall Process of Ordering URLs in both Dynamic and Static Frontiers and Detecting Changes in their Web Pages



(a): Process of Detecting Changes in Web Pages



(b): Process of Creating the Dynamic and Static Frontier



(c): Process of Returning the Highest URL

Figure B.1: Overall Crawling Process of Ordering URLs and Detecting Changes in Web Pages

Appendix C

Parsing Web Pages

WEB PAGE TEXT SEGMENTATION (TOKENIZATION) ==>>
TOTAL NUMBER OF WORDS: ==>> 4856

WEB PAGE TOKENS = { Tennis, BBC, Sport, Accessibility, links, Skip, to, content, Accessibility, Help, BBC, id, Notifications, BBC, navigation, Home, Home, News, News, Sport, Weather, Shop, Earth, Travel, Capital, iPlayer, Culture, Autos, Future, TV, Radio, CBBC, CBeebies, Food, iWonder, Bitesize, Travel, Music, Earth, Arts, Make, It, Digital, Taster, Nature, Local, Tomorrow, s, World, Menu, Search, Search, the, BBC, Search, the, BBC, BBC SPORT, All, Sport, All, Sport, Home, Football, Formula, Cricket, Rugby, U, Tennis, Golf, Athletics, Cycling, BBC, Sport, Home, Football, Formula, Cricket, Rugby, Union, Tennis, Golf, Athletics, Cycling, A, Z, Sports, American, Football, Athletics, Basketball, Boxing, Cricket, Cycling, Darts, Disability, Sport, Football, Formula, Gaelic, Games, Golf, Gymnastics, Horse, Racing, Mixed, Martial, Arts, Motorsport, Olympic, Sports, Rugby, League, Rugby, Union, Snooker, Swimming, Tennis, Winter, Sports, Full, Sports, A, Z, Events, Winter, Olympics, Commonwealth, Games, World, Cup, Around, the, UK, England, Scotland, Wales, Northern, Ireland, World, Sport, European, Football, Champions, League, African, Football, Sport, Africa, Tennis, Live, Scores, Results, Calendar, Order, of, Play, Men, s, Rankings, Women, s, Rankings, Live, Scores, Results, Calendar, Order, of, Play, Men, s, Rankings, Women, s, Rankings, Tennis, Home, Tennis, Top, Stories, Cilic, beats, Pouille, as, Croatia, win, Davis, Cup, final, Marin, Cilic, beat, Lucas, Pouille, in, straight, sets, to, secure, Croatia, s, second, Davis, Cup, title, Nov, Read, more, on, Cilic, beats, Pouille, as, Croatia, win, Davis, Cup, final, France, fight, back, in, Davis, Cup, final, France, keep, their, Davis, Cup, final, hopes, alive, as, Nicolas, Mahut, and, Pierre, Hugues, Herbert, beat, Croatian, pair, Ivan, Dodig, and, Mate, Pavic, Nov, Read, more, on, France, fight, back, in, Davis, Cup, final, Fruitful, talks, on, single, ATP, Cup, and, Davis, Cup, competition, International, Tennis, Federation, president, David, Haggerty, is, quot, confident, quot, a, revamped, Davis, Cup, and, the, new, ATP, Team, Cup, can, join, forces, Nov, Read, more, on, Fruitful, talks, on, single, ATP, Cup, and, Davis, Cup, competition, Croatia, take, control, of, Davis, Cup, final, Marin, Cilic, and, Borna, Coric, both, win, to, put, Croatia, up, in, the, Davis, Cup, final, against, hosts, France, Nov, Read, more, on, Croatia, take, control, of, Davis, Cup, final, Former, mixed, double, champion, Gimmelstob, charged, with, assault, in, LA, Nov, Read, more, on, Former, mixed, double, champion, Gimmelstob, charged, with, assault, in, LA, Bracciali, gets, s, life, ban, for, match, fixing, Starace, also, suspended, Nov, Read, more, on, Bracciali, gets, life, ban, for, match, fixing, Starace, also, suspended, Handstands, and, paddling, without, a, canoe, meet, the, man, making, Murray, normal, again, Nov, From, the, section, Sport, Read, more, on, Handstands, and, paddling, without, a, canoe, meet, the, man, making, Murray, normal, again, Federer, won, t, win, a, Grand, Slam, in, Nov, Read, more, on, Federer, won, t, win, a, Grand, Slam, in, A, new, superstar, has, arrived, Becker, says, Zverev, s, win, is, a, watershed, moment, Nov, Read, more, on, A, new, superstar, has, arrived, Becker, says, Zverev, s, win, is, a, watershed, moment, Zverev, stuns, Djokovic, to, win, ATP, Finals, report, amp, highlights, Nov, Read, more, on, Zverev, stuns, Djokovic, to, win, ATP, Finals, report, amp, highlights, Watch, the, moment, Zverev, stuns, Djokovic, with, sublime, winner, Nov, Read, more, on, Watch, the, moment, Zverev, stuns, Djokovic, with, sublime, winner, Highlights, Dominant, Zverev, beats, Djokovic, to, win, ATP, Finals, Nov, Read, more, on, Highlights, Dominant, Zverev, beats, Djokovic, to, win, ATP, Finals, Hunt, and, Mathewson, lose, in, final, Nov, From, the, section, Disability, Sport, Read, more, on, Hunt, and, Mathewson, lose, in, final, Relive, how, Zverev, beat, Djokovic, to, win, ATP, Finals, Nov, Read, more, on, Relive, how, Zverev, beat, Djokovic, to, win, ATP, Finals, I, hope, the, boy, doesn, t, have, a, sleepless, night, Federer, defends, Zverev, and, ball, boy, Nov, Read, more, on, I, hope, the, boy, doesn, t, have, a, sleepless, night, Federer, defends, Zverev, and, ball, boy, Show, more, Live, tennis, coverage, on, the, BBC, Check, out, the, BBC, Sport, Live, Guide, for, details, of, all, the, forthcoming, live, tennis, on, the, BBC, Read, more, on, Live, tennis, coverage, on, the, BBC, Scores, a, mp, Results, ATP, Tour, CalendarLive, ScoresOfficial, website, WTA, Tour, CalendarLive, ScoresOfficial, website, Audio, amp, Video, Watch, Zverev, beats, Federer, in, ATP, semi, final, Watch, highlights, as, Alexander, Zverev, beats, Roger, Federer, in, a, controversial, semi, final, at, London, s, O, Arena, Read, more, on, Watch, Zverev, beats, Federer, in, ATP, semi, final, Hewitt, and, Kuerten, on, Murray, return, amp, managing, fitness, Read, more, on, Hewitt, and, Kuerten, on, Murray, return, amp, managing, fitness, Watch, Zverev, beats, Isner, to, seal, semi, final, place, best, shots, Read, more, on, Watch, Zverev, beats, Isner, to, seal, semi, final, place, best, shots, Watch, Thiem, s, dominant, display, over, Nishikori, best, shots, Read, more, on, Watch, Thiem, s, dominant, display, over, Nishikori, best, shots, Watch, Djokovic, into, semis, after, win, over, Zverev, Read, more, on, Watch, Djokovic, into, semis, after, win, over, Zverev, Podcast, live, tennis, Djokovic, s, emotions, amp, Murray, s, gymn

Figure C.1: Portion of Tokens of (<http://www.bbc.com/sport/tennis>) Web Page

Table C.1: English Stop Words

Stop word	Stop word	Stop word	Stop word	Stop word	Stop word
a	didn't	him	not	that	we're
about	do	himself	o	that's	we've
above	does	his	of	the	were
after	doesn't	how	off	their	weren't
again	doing	how's	on	theirs	what
against	don't	i	once	them	what's
all	down	i'd	only	themselves	when
am	during	i'll	or	then	when's
an	e	i'm	other	there	where
and	each	i've	ought	there's	where's
any	f	if	our	these	which
are	few	in	ours	they	while
aren't	for	into	ourselves	they'd	who
as	from	is	out	they'll	who's
at	further	isn't	over	they're	whom
b	g	it	own	they've	why
be	h	it's	p	this	why's
because	had	its	q	those	with
been	hadn't	itself	r	through	won't
before	has	j	s	to	would
being	hasn't	k	same	too	wouldn't
below	have	l	shan't	u	x
between	haven't	let's	she	under	y
both	having	m	she'd	until	you
but	he	me	she'll	up	you'd
by	he'd	more	she's	v	you'll
c	he'll	most	should	very	you're
can't	he's	mustn't	shouldn't	w	you've
cannot	her	my	so	was	your
could	here	myself	some	wasn't	yours
couldn't	here's	n	such	we	yourself
d	hers	no	t	we'd	yourselves

Stop word	Stop word	Stop word	Stop word	Stop word	Stop word
did	herself	nor	than	we'll	z

TOTAL NUMBER OF WORDS AFTER ELIMINATING STOP WORDS ==> 351

```

WEB PAGE WORDS AFTER STEMMING = { tenni , sport , access , skip , content , help , notif , navig , news , digit , tomorrow , world , menu , search , search , bbc , foot
bal , formula , cricket , rugbi , golf , athlet , cycl , union , sport , american , basketbal , box , dart , disabl , gaelic , game , gymnast , hors , race , mix , mart
ial , motorsport , olymp , leagu , snooker , swim , winter , full , event , olymp , commonwealth , cup , around , england , scotland , wale , northern , ireland , europ
ean , champion , african , africa , live , score , result , calendar , order , play , men , rank , women , top , stori , cilic , beat , pouill , croatia , win , davi ,
final , marin , beat , luca , straight , set , secur , second , titl , nov , read , franc , fight , back , keep , hope , aliv , nicola , mahut , pierr , hugu , herbert
, croatian , pair , ivan , dodig , mate , pavic , fruit , talk , singl , atp , competit , intern , feder , presid , david , haggerti , quot , confid , revamp , new , te
am , join , forc , take , control , borna , coric , put , host , former , doubl , champion , gimelstob , charg , assault , bracciali , life , ban , match , fix , starac
, suspend , handstand , paddl , cano , meet , man , make , murray , normal , section , feder , won , grand , slam , superstar , arriv , becker , saY , zverev , watersh
, moment , stun , djokov , final , report , amp , highlight , watch , sublim , winner , domin , hunt , mathewson , lose , reliv , hope , boY , doesn , sleepless , nigh
t , defend , ball , show , coverag , check , guid , detail , forthcom , tour , calendarl , scoresoffici , websit , wta , audio , video , semi , alexand , roger , contro
versi , london , arena , hewitt , kuerten , return , manag , fit , isner , seal , place , best , shot , thiem , displaY , nishikori , semi , podcast , emot , embarrass
, defeat , anderson , featur , question , fare , tri , quiz , Year , event , upon , time , test , memori , russel , fuller , look , reform , playeR , much , richer , mi
ght , need , consid , retir , tournament , wimbledon , pat , cash , domin , nadal , bori , tim , henman , serena , william , coach , patrick , mouratoglou , assess , no
vak , challeng , male , playeR , featur , analysi , opinion , correspond , get , fun , fit , cater , level , abil , find , nearest , court , learn , basic , includ , ch
ampionship , open , weekend , big , bash , social , media , latest , headlin , sent , phone , newslett , onlin , explor , rumbl , jungl , replaY , celebr , icon , momen
t , extenn , lta , campbel , appoint , macclesfield , manag , star , driver , mouth , water , palmer , surreY , begin , counti , defenc , essex , individu , ego , selfi
sh , cost , burgess , tomkin , sign , hull , contract , rai , win , mcolgan , urg , detail , euro , plan , feed , use , polici , parent , guidanc , personalis , newsle
tt , choic , respons , site , approach , link , }

```

Figure C.2: Displays Array (A) After Eliminating Stop Words and Performing the Stemming Process

WEB PAGE VECTOR: { [sport, 2], [search, 2], [olymp, 2], [event, 2], [champion, 2], [beat, 2], [win, 2], [final, 2], [hope, 2], [feder, 2], [moment, 2], [domin, 2], [detail, 2], [semi, 2], [manag, 2], [fit, 2], [featur, 2], [player, 2], [newslett, 2], [tenni, 1], [access, 1], [skip, 1], [content, 1], [help, 1], [notif, 1], [navig, 1], [news, 1], [digit, 1], [tomorrow, 1], [world, 1], [menu, 1], [bbc, 1], [footbal, 1], [formula, 1], [cricket, 1], [rugbi, 1], [golf, 1], [athlet, 1], [cycl, 1], [union, 1], [american, 1], [basketbal, 1], [box, 1], [dart, 1], [disabl, 1], [gaelic, 1], [game, 1], [gymmast, 1], [hors, 1], [race, 1], [mix, 1], [martial, 1], [motorsport, 1], [leagu, 1], [snooker, 1], [swin, 1], [winter, 1], [full, 1], [cup, 1], [around, 1], [england, 1], [scotland, 1], [wale, 1], [northern, 1], [ireland, 1], [european, 1], [african, 1], [africa, 1], [live, 1], [score, 1], [result, 1], [calendar, 1], [order, 1], [play, 1], [men, 1], [rank, 1], [women, 1], [top, 1], [stori, 1], [cilic, 1], [pouill, 1], [croatia, 1], [davi, 1], [marin, 1], [luca, 1], [straight, 1], [set, 1], [secur, 1], [second, 1], [titl, 1], [nov, 1], [read, 1], [franc, 1], [fight, 1], [back, 1], [keep, 1], [aliv, 1], [nicola, 1], [mahut, 1], [pierr, 1], [hugu, 1], [herbert, 1], [croatian, 1], [pair, 1], [ivan, 1], [dodig, 1], [mate, 1], [pavic, 1], [fruit, 1], [talk, 1], [singl, 1], [atp, 1], [competit, 1], [intern, 1], [presid, 1], [david, 1], [haggerti, 1], [quot, 1], [confid, 1], [revamp, 1], [new, 1], [team, 1], [join, 1], [forc, 1], [take, 1], [control, 1], [borna, 1], [coric, 1], [put, 1], [host, 1], [former, 1], [doubl, 1], [gimelstob, 1], [chang, 1], [assault, 1], [bracciali, 1], [life, 1], [ban, 1], [match, 1], [fix, 1], [starac, 1], [suspend, 1], [handstand, 1], [paddl, 1], [cano, 1], [meet, 1], [man, 1], [make, 1], [murray, 1], [normal, 1], [section, 1], [won, 1], [grand, 1], [slam, 1], [superstar, 1], [arriv, 1], [becker, 1], [sav, 1], [zverev, 1], [watersh, 1], [stun, 1], [djokov, 1], [report, 1], [amp, 1], [highlight, 1], [watch, 1], [sublim, 1], [winner, 1], [hunt, 1], [mathewson, 1], [lose, 1], [reliv, 1], [bov, 1], [doesnt, 1], [sleepless, 1], [night, 1], [defend, 1], [ball, 1], [show, 1], [coverag, 1], [check, 1], [guid, 1], [forthcom, 1], [tour, 1], [calendar, 1], [websit, 1], [wta, 1], [audio, 1], [video, 1], [alexand, 1], [roger, 1], [london, 1], [arena, 1], [hewitt, 1], [kuerten, 1], [return, 1], [isner, 1], [seal, 1], [place, 1], [best, 1], [shot, 1], [thiem, 1], [display, 1], [nishikori, 1], [podcast, 1], [emot, 1], [embarrass, 1], [defeat, 1], [anderson, 1], [question, 1], [fare, 1], [tri, 1], [quiz, 1], [Year, 1], [upon, 1], [time, 1], [test, 1], [memori, 1], [russel, 1], [fuller, 1], [look, 1], [reform, 1], [much, 1], [richer, 1], [might, 1], [need, 1], [consid, 1], [retir, 1], [tournament, 1], [wimbledon, 1], [pat, 1], [cash, 1], [nadal, 1], [bori, 1], [tim, 1], [henman, 1], [serena, 1], [william, 1], [coach, 1], [patrick, 1], [assess, 1], [novak, 1], [challeng, 1], [male, 1], [analysi, 1], [opinion, 1], [correspond, 1], [get, 1], [fun, 1], [cater, 1], [level, 1], [abil, 1], [find, 1], [nearest, 1], [court, 1], [learn, 1], [basic, 1], [includ, 1], [open, 1], [weekend, 1], [big, 1], [bash, 1], [social, 1], [media, 1], [latest, 1], [headlin, 1], [sent, 1], [phone, 1], [onlin, 1], [explor, 1], [rumbl, 1], [jungl, 1], [replay, 1], [celebr, 1], [icon, 1], [extern, 1], [lta, 1], [campbel, 1], [appoint, 1], [star, 1], [driver, 1], [mouth, 1], [water, 1], [palmer, 1], [surrey, 1], [begin, 1], [counti, 1], [defenc, 1], [essex, 1], [individu, 1], [ego, 1], [selfish, 1], [cost, 1], [burgess, 1], [tomkin, 1], [sign, 1], [hull, 1], [contract, 1], [rai, 1], [mccolgan, 1], [urg, 1], [euro, 1], [plan, 1], [feed, 1], [use, 1], [policci, 1], [parent, 1], [guidanc, 1], [personalis, 1], [choic, 1], [respons, 1], [site, 1], [approach, 1], [link, 1], }

Figure C.3: Illustrates the Vector of (<http://www.bbc.com/sport/tennis>) Web Page

Appendix D

Terms Selection

Table D.1: Pages Used in Terms Extraction

Category	Web Pages
Business	https://www.wsj.com/india https://www.ft.com/ https://money.cnn.com/ https://www.foxbusiness.com/ https://www.morningstar.com/ http://fortune.com/ https://www.business-standard.com/ https://www.thehindubusinessline.com/ https://www.businesstoday.in/ https://business.financialpost.com/ https://www.moneyweb.co.za/ https://www.businessdailyafrica.com/ https://www.baystreet.ca/ https://smallcappower.com/ https://economictimes.indiatimes.com/
Crime	http://www.insidesocal.com/sgvcrime/ https://mylifeofcrime.wordpress.com/ https://blogs.findlaw.com/blotter/ http://www.crime-research.org/latestnews https://www.crimetraveller.org/ https://www.reddit.com/r/TrueCrime http://spdblotter.seattle.gov/ http://www.cwbchicago.com/ https://nypost.com/tag/crime/ http://www.blog.murdermap.co.uk/ https://www.thecrimemag.com/ https://ididitforjodie.com/ http://ukcriminallawblog.com/ https://the-line-up.com/ http://crimedaily.com/ https://www.huffingtonpost.com/section/crime
Health	https://consumer.healthday.com/ https://www.bbc.com/news/health https://www.msn.com/en-us/health/ https://edition.cnn.com/health https://www.huffingtonpost.com/section/healthy-living https://www.reuters.com/news/health https://www.telegraph.co.uk/health/ https://www.ctvnews.ca/health https://www.yahoo.com/lifestyle/tagged/health https://www.drugs.com/ https://www.everydayhealth.com/ https://www.healthline.com/ https://www.mercola.com/ https://www.mindbodygreen.com/ https://www.rxlist.com/script/main/hp.asp https://www.medicalnewstoday.com/

Category	Web Pages
Politics	https://www.politico.com/ https://www.dailykos.com/ https://www.reuters.com/politics https://www.infowars.com/ https://www.theblaze.com/ https://www.wnd.com/ https://www.newsmax.com/ https://www.dailywire.com/ https://www.westernjournal.com/ct/ https://www.theamericanconservative.com/ https://abcnews.go.com/ https://www.factcheck.org/ https://www.reddit.com/r/politics https://hotair.com/ https://talkingpointsmemo.com/ https://www.redstate.com/
Sports	https://sports.yahoo.com/ http://www.espn.com/ http://www.espncricinfo.com/ https://www.cbssports.com/ http://www.skysports.com/ https://www.nbcsports.com/ https://www.foxsports.com/ https://www.tsn.ca/ http://www.sportingnews.com/ https://www.sportskeeda.com/ https://www.yardbarker.com/ https://www.mlb.com/news http://www.secsports.com/ https://www.sportstarlive.com/ https://www.reddit.com/r/sports/ https://thebiglead.com/
Technology	https://www.firstpost.com/tech https://www.techradar.com/ https://www.bbc.com/news/technology https://blog.us.playstation.com/ https://www.engadget.com/ https://www.reuters.com/news/technology https://www.slashgear.com/ https://www.droid-life.com/ https://abcnews.go.com/technology https://arstechnica.com/ https://website-designs.com/ http://www.freshnews.org/ https://www.theinquirer.net/ https://www.techdirt.com/

Table D.2: Irrelevant URLs of Crime News Category

URL	Similarity
http://www.aljazeera.com/news/2018/04/50-years-mlk-death-youth-living-legacy-180404085546543.html	0.11009
http://www.aljazeera.com/news/2018/11/white-grocery-store-shooting-suspect-charged-hate-crimes-181116005521961.html	0.11712
http://www.aljazeera.com/news/2018/06/philippine-catholic-priests-killing-180613073921062.html	0.11523
http://www.aljazeera.com/news/2018/07/philippine-mayor-antonio-halili-assassinated-ceremony-180702054409964.html	0.11738
http://www.aljazeera.com/news/2016/12/pushes-murder-probe-rodrigo-duterte-161221043326833.html	0.12986
http://www.aljazeera.com/news/2018/02/florida-high-school-shooting-180215061404114.html	0.1043
http://www.aljazeera.com/news/2018/11/suspect-dead-multiple-injured-chicago-hospital-shooting-181119230706736.html	0.13068
http://www.aljazeera.com/indepth/interactive/2016/08/philippines-death-toll-duterte-war-drugs-160825115400719.html	0.10524
http://www.aljazeera.com/news/2018/06/capital-gazette-victims-remembered-edition-attack-180629105203859.html	0.11214
http://www.aljazeera.com/news/2018/04/duterte-icc-prosecutor-arrest-180413053920297.html	0.12182
http://www.aljazeera.com/news/2018/10/pittsburgh-synagogue-shooting-suspect-charged-44-counts-181031174612815.html	0.10608
http://www.aljazeera.com/news/2017/07/joko-widodo-police-shoot-suspected-drug-dealers-170722104559016.html	0.1363
http://www.aljazeera.com/news/2018/11/australia-police-arrest-men-terrorist-attack-plan-181120003913960.html	0.1142
http://www.aljazeera.com/programmes/aljazeeraworld/2018/11/killed-robert-kennedy-181112080415326.html	0.10379
http://www.aljazeera.com/news/2018/10/india-whatsapp-murders-phone-rumours-spark-frenzied-mobs-181024104247089.html	0.10204
http://www.aljazeera.com/news/2017/03/duterte-international-criminal-court-stop-170319134647429.html	0.14643
http://www.aljazeera.com/news/2017/08/80-killed-escalation-philippines-war-drugs-170818042429248.html	0.1098
http://www.aljazeera.com/news/2017/09/entire-philippine-city-police-force-fired-killings-170916025615936.html	0.14888

Appendix E

Reading Robot File

```
User-agent: Googlebot
Sitemap: http://www.bbc.com/sitemap.xml
Sitemap: http://www.bbc.com/sitemaps/index-com-news.xml
Sitemap: http://www.bbc.com/sitemaps/index-com-archive.xml
Sitemap: http://www.bbc.com/video_sitemap.xml
Disallow: /iplayer/episode/*?from=r*
Disallow: /iplayer/cy/episode/*?from=r*
Disallow: /iplayer/gd/episode/*?from=r*
Disallow: /_programmes
Disallow: /606/
Disallow: /aboutthebbc/insidethebbc/search
Disallow: /academy/chinese-trad/search
Disallow: /afrique/search
Disallow: /apps/cbbc
Disallow: /apps/flash
Disallow: /apps/ide
Disallow: /apps/ids
Disallow: /apps/ifl
Disallow: /apps/nr
Disallow: /apps/query
Disallow: /apps/t
```

} blocked URLs of Googlebot

(a): A Collection of Blocked URLs of Googlebot Agent

```
User-agent: *
Sitemap: http://www.bbc.com/sitemap.xml
Disallow: /_programmes
Disallow: /606/
Disallow: /aboutthebbc/insidethebbc/search
Disallow: /academy/chinese-trad/search
Disallow: /afrique/search
Disallow: /apps/cbbc
Disallow: /apps/flash
Disallow: /apps/ide
Disallow: /apps/ids
Disallow: /apps/ifl
Disallow: /apps/nr
Disallow: /apps/query
Disallow: /apps/t
Disallow: /arabic/search
Disallow: /arts/yourpaintings/artdetective/
Disallow: /arts/yourpaintings/feedback/sent/
```

} blocked URLs of all user_agent, but will neglected by Googlebot and other specified agent in robot file

(b): A Collection of Blocked URLs of all Agents

Figure E.1: A Portion of Blocked URLs of User Agents in (*www.bbc.com*) Robot File

```

.....
Line = # www.bbc.com
Command = COMMENT - URL =
.....
Line = User-agent: Googlebot
Command = user-agent - URL =
.....
Line = Sitemap: http://www.bbc.com/sitemap.xml
Command = sitemap - URL = http://www.bbc.com/sitemap.xml
.....
Line = Sitemap: http://www.bbc.com/sitemaps/index-com-news.xml
Command = sitemap - URL = http://www.bbc.com/sitemaps/index-com-news.xml
.....
Line = Sitemap: http://www.bbc.com/sitemaps/index-com-archive.xml
Command = sitemap - URL = http://www.bbc.com/sitemaps/index-com-archive.xml
.....
Line = Sitemap: http://www.bbc.com/video_sitemap.xml
Command = sitemap - URL = http://www.bbc.com/video_sitemap.xml
.....
Line = Disallow: /iplayer/episode/*?from=r*
Command = disallow - URL = /iplayer/episode/*?from=r*
BLOCKED URL: /iplayer/episode/*?from=r* FOR OTHER USER_AGENT
URL WILL NOT BE ADDED TO BLOCKED LIST
.....
Line = Disallow: /iplayer/cy/episode/*?from=r*
Command = disallow - URL = /iplayer/cy/episode/*?from=r*
BLOCKED URL: /iplayer/cy/episode/*?from=r* FOR OTHER USER_AGENT
URL WILL NOT BE ADDED TO BLOCKED LIST
.....
Line = Disallow: /iplayer/gd/episode/*?from=r*
Command = disallow - URL = /iplayer/gd/episode/*?from=r*
BLOCKED URL: /iplayer/gd/episode/*?from=r* FOR OTHER USER_AGENT
URL WILL NOT BE ADDED TO BLOCKED LIST
.....
Line = Disallow: /_programmes
Command = disallow - URL = /_programmes
BLOCKED URL: /_programmes FOR OTHER USER_AGENT
URL WILL NOT BE ADDED TO BLOCKED LIST
.....
Line = Disallow: /606/
Command = disallow - URL = /606/
BLOCKED URL: /606/ FOR OTHER USER_AGENT
URL WILL NOT BE ADDED TO BLOCKED LIST
.....

```

comment

user_agent: Googlebot

sitemaps

blocked URLs for Googlebot agent

(a): A Portion of Blocked URLs of Googlebot User_agent

```

.....
Line = User-agent: *
Command = user-agent - URL =
* - Rules apply
.....
Line = Sitemap: http://www.bbc.com/sitemap.xml
Command = sitemap - URL = http://www.bbc.com/sitemap.xml
.....
Line = Disallow: /_programmes
Command = disallow - URL = /_programmes
BLOCKED URL: /_programmes
URL ADDED TO BLOCK LIST
.....
Line = Disallow: /606/
Command = disallow - URL = /606/
BLOCKED URL: /606/
URL ADDED TO BLOCK LIST
.....
Line = Disallow: /aboutthebbc/insidethebbc/search
Command = disallow - URL = /aboutthebbc/insidethebbc/search
BLOCKED URL: /aboutthebbc/insidethebbc/search
URL ADDED TO BLOCK LIST
.....
Line = Disallow: /academy/chinese-trad/search
Command = disallow - URL = /academy/chinese-trad/search
BLOCKED URL: /academy/chinese-trad/search
URL ADDED TO BLOCK LIST
.....
Line = Disallow: /afrique/search
Command = disallow - URL = /afrique/search
BLOCKED URL: /afrique/search
URL ADDED TO BLOCK LIST
.....

```

user_agent: *

sitemap

blocked URLs for all user_agent

(b): A Portion of Blocked URLs for All User-agents

Figure E.2: Results of Extracting Blocked URLs of (www.bbc.com) Robot File

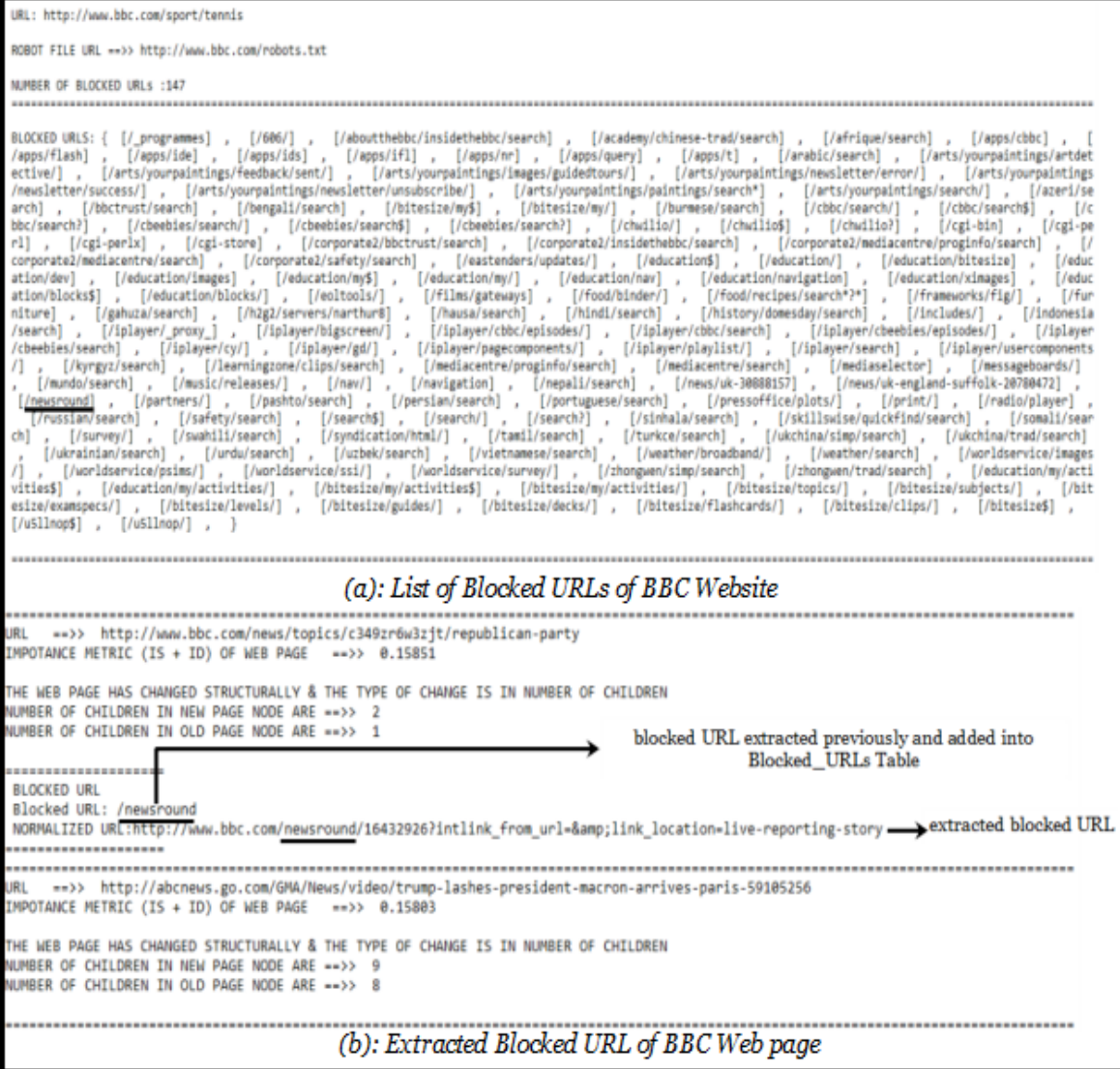


Figure E.3: Extracting and Detecting Blocked URLs

Appendix F

The Frontier



Figure F.1: Portion of the Dynamic and Static URLs of Political News Category

```

URL ==>> http://www.theguardian.com/us-news/us-midterm-elections-2014
IMPOTANCE METRIC (IS + ID) OF WEB PAGE ==>> 0.24593

WEB PAGE HAS NOT CHMAGED
=====
URL ==>> http://www.bbc.com/news/av/world-us-canada-37913568/election-2016-trump-and-clinton-cast-their-votes-early
IMPOTANCE METRIC (IS + ID) OF WEB PAGE ==>> 0.24129

WEB PAGE HAS NOT CHMAGED
=====
URL: http://www.theguardian.com/us-news/live/2018/nov/20/trump-latest-live-news-updates-ivanka-emails-asylum-ruling-cnn-us-politics-today

SIMILARITY METRIC OF URL ANCHOR TEXT ==>> 0.2394737360357
SIMILARITY METRIC OF WEB PAGE TEXT ==>> 0.240410490745004
AVERAGE OF SIMILARITY METRIC (IS) ==>> 0.239942113390352
WEB PAGE CATEGORY ==>> POLITICS

WEB PAGE ACCEPTED AND STORED
=====
URL: http://www.theguardian.com/us-news/video/2018/oct/27/barack-obama-trump-and-other-republicans-just-making-stuff-up-video

SIMILARITY METRIC OF URL ANCHOR TEXT ==>> 0.2394737360357
SIMILARITY METRIC OF WEB PAGE TEXT ==>> 0.0937958099221084
AVERAGE OF SIMILARITY METRIC (IS) ==>> 0.16663472978904
WEB PAGE CATEGORY ==>> POLITICS

WEB PAGE ACCEPTED AND STORED
=====
URL ==>> http://www.theguardian.com/us-news/us-midterm-elections-2010
IMPOTANCE METRIC (IS + ID) OF WEB PAGE ==>> 0.23646

TEXTUAL CHMAGE:
NODE TEXT IN NEW PAGE ==>> Tea Party movement
NODE PATH IN NEW PAGE ==>> /html[1]/body[1]/div[4]/div[1]/section[16]/div[1]/div[1]

NODE TEXT IN OLD PAGE ==>> US Congress
NODE PATH IN OLD PAGE ==>> /html[1]/body[1]/div[4]/div[1]/section[16]/div[1]/div[1]/ul[1]/li[5]/a[1]/#text[1]
=====

```

Figure F.2: Crawling New URLs Before Pre-stored URLs

Appendix G

Terms of News Categories

Table G.1: Terms of News Categories

Business	Crime	Health	Politics	Sport	Technology
market	case	health	minist	team	review
invest	traffick	news	prime	game	app
compani	crime	help	trump	sport	report
report	investig	drug	elect	plaYer	game
busi	murder	cancer	report	report	featur
stock	report	report	democrat	win	launch
news	news	diseas	presid	score	news
fund	offic	medic	state	leagu	facebook
financi	polic	diet	vote	make	appl
power	suspect	food	polit	open	site
trade	arrest	caus	senat	news	tech
plan	victim	new	polic	live	updat
bank	releas	top	news	plaY	phone
economi	homicid	studi	republican	schedul	charg
manag	detect	risk	offic	time	secur
investor	convict	heart	american	show	googl
price	person	prevent	candid	footbal	compani
industri	death	treat	gop	final	android
buy	killer	eat	nation	nfl	technolog
billion	court	research	govern	season	network
oil	accus	doctor	america	basketbal	plaYstat
sale	drive	patient	campaign	basebal	amazon
rise	fire	hospit	investig	hockeY	platform
interest	alleg	price	nuclear	box	iphon
deal	reason	world	midterm	soccer	microsoft
paid	prison	cost	world	tenni	twitter
produc	car	life	lead	golf	privaci
car	dead	effect	clinton	athlet	protect
navig	miss	treatment	obama	chess	samsung
data	shoot	scienc	russian	swim	design
entrepre- neurship	assault	compani		coach	releas

Business	Crime	Health	Politics	Sport	Technology
servic	drug	medicin		minut	robot
boost	disappear	attack		retir	attack
import	kidnap	care		health	mobil
respons	justic	clinic		race	Youtub
china	shot			champion	websit
global	die			adventur	smartphon
	law			dive	tweet
	plead			match	laptop
	sheriff			cricket	netflix
	driver			sail	camera
	injuri			beat	machin
	polici			olymp	soni
	attack			cup	spotifi
	stori			cycl	huawei
	search			challeng	plan
	life			rugbi	data
	podcast			fifa	breach
	face			champ	polit
	crimin			nba	intern
	launder			winner	media
	gun				system
	guilti				elon
	violent				musk
	appeal				tesla
	jail				hack
	threaten				photograph
	cop				devic
	reserv				car
	kill				electr
	man				telecom
	Girl				Yahoo
	corrupt				comput
	right				
	human				
	fault				
	radicalis				

Business	Crime	Health	Politics	Sport	Technology
	survivor				
	pictur				
	khashoggi				
	wit				
	Youth				
	head				