Middle East University for Graduate Studies

Faculty of Information Technology
Department of Computer Information Systems

A Hybrid Approach for Web Change Detection

Thesis submitted in partial Fulfillment of the Requirements for Master Degree in Computer Information Systems

By
Sakher Khaleel Al-Qaaideh

Supervisor

Professor Mohammad A. Al-Fayoumi (PhD)
Faculty of Information Technology
Middle East University for Graduate Studies

Co-Supervisor

Dr. Ezz Hattab
Associate Professor in e-Business
The Arab Academy for Banking and Financial Sciences

February - 2009

Authorization

I am Sakher Khaleel Al-Qaaideh , authorize the Middle East University for Graduate Studies to supply copies of this thesis to the library or establishments or individuals upon request.

Signature:

4/2/2009

تفويض

انا صخر خليل القعايدة افوض جامعة الشرق الاوسط للدراسات العليا بتزويد نسخ من هذه الرسالة الى المكتبة او المؤسسات او الاشخاص عند طلبها.

التوقيع:

2009/2/4

Committee Decision

This thesis "A Hybrid Approach for Web Change Detection" was successfully defended and proved on February 4th 2009.

Examination Committee

Prof. Mohammad A. Al-Fayoumi Signature

Dr. Ezz Hattab Signature

Dr. Musbah Aqel Signature

Dr. Raad Elwan Signature

Dedication

To my mother, who raised the creativity in me…
To my father, who supported and encouraged me…

To my wife, Sana, for being the origin of my inspiration…
To my brothers and sisters, who motivated me…

Acknowledgements

I would like to thank God for His guidance in my life.

I would like to thank the professors who supervised this thesis, Prof. Mohammad Al-Fayoumi and Dr. Ezz Hattab for their unconditional support and guidance in helping me write this thesis.

My appreciation to Information Technology faculty instructors and staff at the Middle East University for Graduate Studies, for their contributions in running this great academic establishment.

I would like to thank my colleagues at the Middle East University for Graduate Studies, for the wonderful memories we lived together.

List of Figures

Figure 1.1: Change rate detection problem 2
Figure 1.2: Mismatches period between search engines and Web servers 3
Figure 1.3: Typical (Ordinary) vs. Meta-updates crawlers (freshness average) 5
Figure 1.4: Conceptual diagram of the problem 6
Figure 1.5: Several options for the synchronization points 7
Figure 2.1: High-level architecture of a standard Web crawler 11
Figure 2.2: Expected ChangeRatio for various sample sizes 18
Figure 2.3: Algorithm of the adaptive-sampling policy 20
Figure 2.4: Downloading pages under low resources 21
Figure 3.1: Illustration of the proposed policy 23
Figure 3.2: Page change rate calculation 26
Figure 3.3: The proposed policy crawling cycle 30
Figure 4.1: Web sites change rate variation over time 34
Figure 4.2: Real change rate using the different policies 36

List of Tables

Table 2.1: Notations used in proportional policy equation 16
Table 2.2: Notations used in sample size equation 19
Table 3.1: Notations used in sample size calculations for the greedy policy 28
Table 4.1: Sample web sites, their real change rate and number of pages 34
Table 4.2: Change-rate detection using different policies and cases 37

الملخص

تقوم محركات البحث بحفظ نسخ من صفحات الانترنت لتسهل على المستخدمين البحث عن المعلومات في هذه الصفحات. تقنيات تعديل وانشاء صفحات الانترنت اصبحت اكثر سهولة واصبح متاحاً للجميع تعديل او انشاء الصفحات, لا تقوم الخوادم التي تحتوي صفحات الانترنت بارسال التعديلات الجديدة الى محركات البحث لانها غير ملزمة بذلك, وبذلك يصبح جزء من الصفحات المحفوظة في محركات البحث غير مطابقة للمصدر, لذا يجب على محرك البحث ان يقوم بحفظ الاصدارات الجديدة من الصفحات التي خضعت للتعديل حتى يحافظ على ميزة تقديم خدمات البحث عن المعلومات المطابقة لما هو موجود في الصفحات الفعلية ولا تصبح المعلومات الموجودة في محرك البحث قديمة مثل الاخبار.

بعض الحلول لهذه المشكلة تقترح ان يتوقع محرك البحث موعد التغير في محتوى الصفحات ليحفظ نسخ منها بعد حدوث التغير مباشرة وليتجنب حفظ نسخ من صفحات اخرى لا تحتوي اي تغييرات جديدة. التغير في الصفحة يمكن يتوقعه عن طريق مراقبة التغيرات خلال الوقت وتحديد معدل التغير لها. طريقة اخرى لمعرفة معدل التغير لمواقع الانترنت تقوم بأخذ عينة من كل موقع الذي يحتوي صفحات الانترنت وتحديد معدل التغير له بناءاً على معدل التغير في العينة, ثم توزع جهود محرك البحث في حفظ نسخ من الصفحات على المواقع تبعاً لمعدل تغيرها. طريقة اخرى تجمع كل صفحات الانترنت من مختلف المواقع في مجموعات حسب معدل التغير لها, ثم تأخذ عينات من كل مجموعة لتحدد معدل التغير لكل مستوى.

قمنا في هذه الدراسة بتطوير طريقة اخرى لتحديد معدل التغير بشكل افضل في صفحات الانترنت عن طريق دمج طريقتان من الطرق المذكورة سابقا، وهما طريقة مراقبة التغيرات في الصفحة لتحديد معدل تغيرها و طريقة اخذ عينة من الموقع لتحديد معدل التغير فيهً. الطريقة الجديد مبنية على أخذ عينة من كل موقع لتحدد معدل التغير له, ثم تراقب التغير في كل صفحة في كل عينة خلال الوقت وخلال نسخ عدة لها حتى تحدد معدل التغير للعينة بشكل ادق أكثر. الاختبارات اثبتت فعالية هذه الطريقة في حالات مختلفة يتعرض لها محرك البحث مثل قلة الامكانيات المتاحة لاخذ عينة من المواقع.

Abstract

Search engines save copies of web pages to facilitate searching for information in these pages. Editing and creating pages techniques become easier and available for everyone to edit or create web pages. Web servers that contain web pages don't submit new changes to search engines because they are not committed to do this. A part of web pages stored in search engine's repository become different than the original source, so search engines should save the new versions of the edited pages to keep the advantage of providing search services of information to be identical to the original source to avoid considering information in search engine old like news.

Some solutions of this problem suggest that the search engine expect the date of the change in pages' content to store them after change occurrence directly and avoid saving other pages that don't contain any new changes. Change in page can be expected by monitoring changes over time and detecting its change rate. Another approach to determine web site change rate is to take sample from each web site that contain web pages and detect change rate of each web site according to change rate in each sample, then distribute search engine efforts for saving pages over web sites according to its change rate. Another approach clusters all web pages from different web sites in clusters of change rate levels, samples pages from each cluster to determine its change rate.

This thesis presents another approach to better change rate detection in web pages by combining two approaches mentioned above, which are monitoring changes in each page to determine its change rate and the other approach is to take sample form each web site to determine its change rate. The new approach is based on sampling pages from web sites, then monitoring changes in each page in the sample over time and over many versions of the page, so we detect change rate of the sample more accurately. Experiments proved the effectiveness of the new approach in different search engine cases like low sources for sampling web sites.

Table of Contents

Chapter 1: Introduction 1
1.1 Background 1
1.2 Problem Statement 2
1.3 Related work 3
1.4 Contributions 9
1.5 Outline 9

Chapter 2: Sampling Policies 10
2.1 Search Engines 10
2.2 Change Detection using Sampling 12
2.3 Why sampling 13
2.4 Measuring the Effectiveness of the Sampling-Policies 13
2.4.1 Introduction 13
2.4.2 ChangeRatio 13
2.4.3 Freshness and Age 14
2.4.4 Divergence 15
2.5 Main Sampling Policies 15
2.5.1 Introduction 15
2.5.2 Proportional Policy 15
2.5.3 Greedy Policy 16
2.5.4 Greedy Policy versus Proportional Policy 17
2.6 Sample Size 17
2.7 Adaptive Sampling 19
2.8 Low Download Resources 19

Chapter 3: The Proposed Policy 22
3.1 Change Frequency-Based and Greedy Sampling Policies 22
3.2 Requirements Analysis 23
3.3 Policy Design 24
3.3.1 First Sample Size 24
3.3.2 Page History Tracking 24
3.3.2.1 Flexible Number of Page Versions 24
3.3.2.2 Page Versions Storage and Indexing 27

3.3.2.3 Change Detection 27
3.3.3 Greedy sampling Implementation 28
3.4 Policy Implementation 29

Chapter 4: Experiments 31
4.1 Used Tools 31
4.2 Web Sample 32
4.3 Greedy-Sampling Policy Implementation 34
4.4 Proposed policy Implementation 35

Chapter 5: Conclusions and Future Work 38
5.1 Conclusions 38
5.2 Future work 39

References 40

Appendices A1
Appendix A: Tools code A1
A.1 The Proposed Approach Sample Calculation Algorithm A1
A.2 The Greedy Sample Calculation Algorithm A2
Appendix B: Curriculum Vitae B1

Chapter 1

Introduction

1.1 Background

Internet has become a part of our life; it becomes the easiest way to get information. New network technologies made reaching the internet easier and cheaper, like digital subscriber line (DSL), most of work places and homes got this data line, and this made the internet available any time. Anyone who uses the internet has discovered the way to reach any needed information, and the first knowledge base we think about when we need to know about something is the internet since it is available, easy, and not costly. In addition to enrollment of internet in the learning activities in schools and academic institutions, it has become the main source of information after completing the study.

In addition to providing information, internet has got its audience by the services it provides like communications and entertainment, which lead to usage by users of different ages. It also forced many to learn computer skills for work requirements after enrollment of the internet in business, where this type of people who heard and watched the creation of computers and the information technology evolution, thought they are not required to get involved in the computer knowledge since it would not be necessary in their life time, but they found themselves forced to learn about computer and internet to keep performing the same activities they used to do in the past.

Technically internet can be used in many ways. The most easily and flexible used way is the web [4], which consists of web sites reached and identified by its domain names, each web site contains a number of pages written using HTML [14] language stands for Hyper Text Markup Language. Web pages support many of other internet services, because most of the efforts focused on developing the web after it proved its excellence on other internet services when these services were presented over web pages, example: Google launched its virtual globe, map and geographic information program Google earth. Most users of this program used it to get maps of cities and roads, after that Google provided the same maps over web on the web site called Google maps. The audience of Google earth program moved to get the same service on the maps web site.

Information retrieving over web became easier after creating search engines [15] which are programs or scripts that have the ability to index large amount of web pages.

Web crawler [3] is one program used in the search engine. It is responsible for visiting web pages for taking copies of them. It organizes these visits to achieve better performance. Search engines are used by its web interfaces, like the most popular search engine Google.

As search engine copies and stores large amount of web pages, many issues came up like how to manage this large amount of data, one of these issues is how to keep the copies of pages identical to the original source, since original pages change independently. In this thesis this problem will be studied with its previous solutions, some of these solutions will be implemented and new approach to solve this problem will be presented and implemented for comparison with pervious implementations.

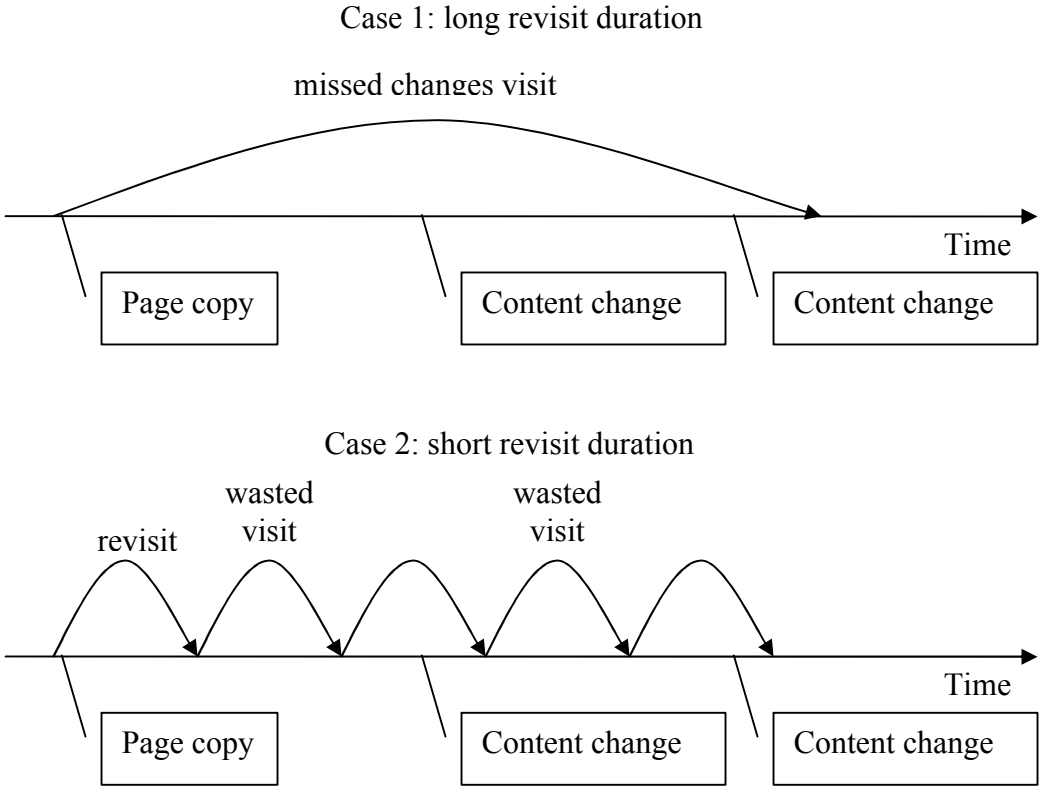


Figure (1.1): Change rate detection

1.2 Problem statement

Web is dynamic; page's content can be edited easily using many tools, search engine's copies of the crawled pages may become old and need to be updated since the crawled page's content changes always, web crawler should detect the change rate to perform

better and utilize its effort. Visiting some pages, founding that its content has not been changed and taking the same stored copy of the page is some kind of web crawler resources wasting. Taking a copy of page's content after many changes occurred in content in the duration between the visits, is an evidence of low search engine performance since searchers will not find the required information in the page at the time when that information is needed. Figure (1.1) illustrates the problem.

1.3 Related work

Many ideas came up to solve the issue of detecting change rate of web pages and keeping the copies of pages fresh as much as we can. Some related work with different policies used will be described in this section, but the last one which talks about sampling policies will be the main inspiration in this study to find new policy due to the agility of these policies and effectiveness which will appear in the experiments.

1.3.1 Ezz hattab, Updating search engines using meta-updates, 2007

The study [11] described the problem of keeping the copy of web pages fresh in search engines, it described how people may not find the requested information in the search engine during the period when change has occurred and no fresh copy has been taken as used figure (1.2).

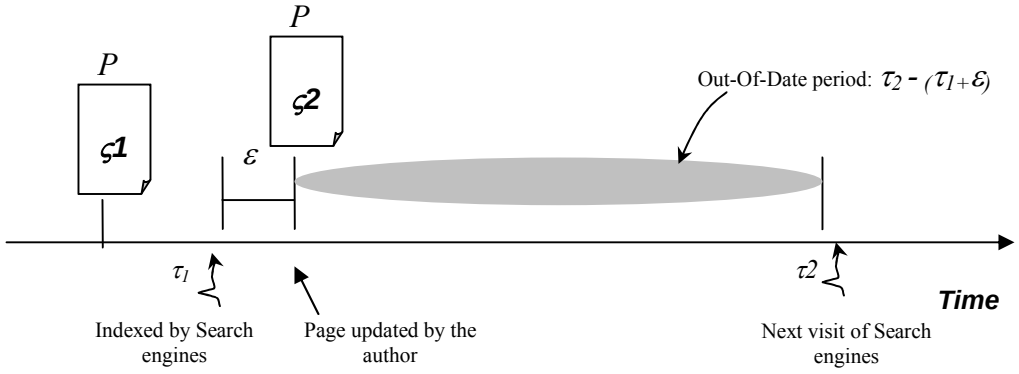


Figure (1.2): Mismatches period between search engines and Web servers

Based on the fact that no web crawler can index the entire web due the huge number of web pages which is approximately a billion pages [13], high grow speed of the web, the required period to re-index all the web which is 8.5 days and assuming that a

web page has an average size of 12 kilobytes, a simple equation has been proposed to calculate the speed of any web crawler that indexes all the web as follow:

$$\frac{10^9}{8.5 days} \times \frac{12 kbytes}{1 page} = 1.3 Gbit/\text{sec}$$

The study showed that this speed is impractical due to the grow speed of the web which is doubling the number of web pages yearly [19]. This scenario suggests getting some assistance by web servers to avoid crawling all the pages in the web every time checking for changes. This help can be done by adding Meta tags to the page which tell the web crawler when to re-index the page again, this will help in decreasing the number of web pages going to be re-indexed by the crawler every time since some of pages do not require a re-indexing because they informed the web crawler that no change has occurred in the content during the period given in the Meta tag. The study proposed the following general form of the Meta tag:

<META NAME="*attribute*" CONTENT="*value*">

"Change-Frequency"
"STATIC" |
"PERIODIC = *Number-Of-Days*"
"non-uniform= "*The-next-change*"

As described above an attribute should contain the following value: Change Frequency-Based and the content should contain 3 types of values to inform the crawler with the change rate of the page:

1) Static value means the page has a little probability to be changed; the study recommends the web crawler to assume that this value is 90 days.
2) Periodic value means the page change in uniform rate, example: PERIODIC=7, means the page change every 7 days.
3) A variable means the page change in a non-uniform rate; this value is generated by the web server which runs dynamic web pages not written by page author.

A web technology called sitemap is similar to this solution, sitemap is an XML file that lists URLs for a site along with additional metadata about each URL (when it was last updated, how often it usually changes, and how important it is, relative to other URLs in the site) so that search engines can more intelligently crawl the site [7]. The study proposed a solution of the problem similar to the sitemap but in different easier

way, because sitemap generating requires extra tools, knowledge of the XML language, extra space on the web server and extra added files, where this solution requires none of these, web page author will have to add the described short HTML tag only.

In the experiments of this study a set of well known web sites have been used to improve the solution, the used crawler crawled the web sites for 3 weeks one time using the Meta tags and the other time without using them. The measure of improvement was the number of needless crawling processes, and the proposed solution showed good results, which can be noticed in figure (1.3) of visual comparison between the typical crawling and the crawling that supports the Meta-updates This chart shows a comparison of the two ways.

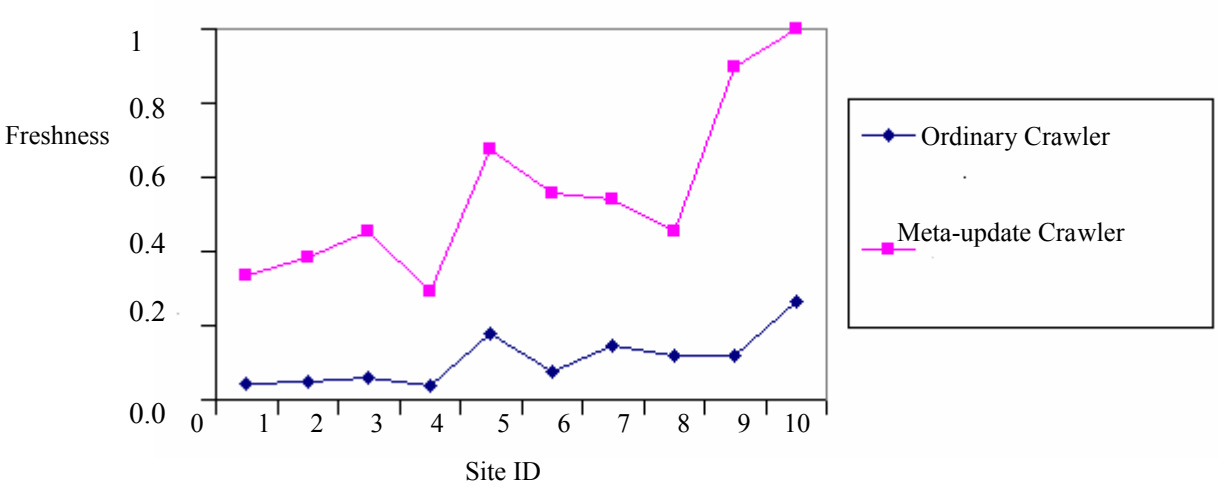


Figure (1.3): Typical (Ordinary) vs. Meta-updates crawlers (freshness average)

The major weak point in this proposed solution and sitemaps is the commitment of web page author of specifying the change rate of the page because web pages can be published without adding Meta-tags or adding details in the sitemap, and no page author is required to do this. The new technologies used in web servers which help in generating dynamically web pages increased the weakness of these solutions, because many web site users involved in creating pages with topics or personal profiles that make it hard to know when its content is going to change, and the goal of ease of creating web pages skipped the step of determining a change rate of the created page. This type of dynamically generated pages makes it harder in selecting web sites for experiments in this study as will be described later in chapter 4.

1.3.2 Junghoo Cho, Hector Garcia-Molina, Synchronizing a Database to Improve Freshness, 2000

This study relied on tracking the change history of pages to detect change rate by finding a policy that implements this, the study defined the policies that perform by the process of making the local copies of web pages up-to-date using change history tracking as synchronization policies, the study defined two freshness metrics, change models of the underlying data, and synchronization policies. It validated the experiments based on pages collected from 270 web sites for more than 4 months, used them also to improve freshness by the new proposed policy.

In the beginning, the study showed how pages change independently, and defined the web crawler as a client of the web site that uses its pages for retrieving copies of them; these pages change without pushing the changes to the client. This case is described in Figure (1.4).

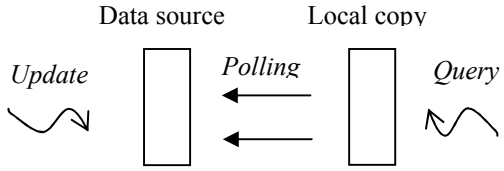


Figure (1.4): Conceptual diagram of the problem

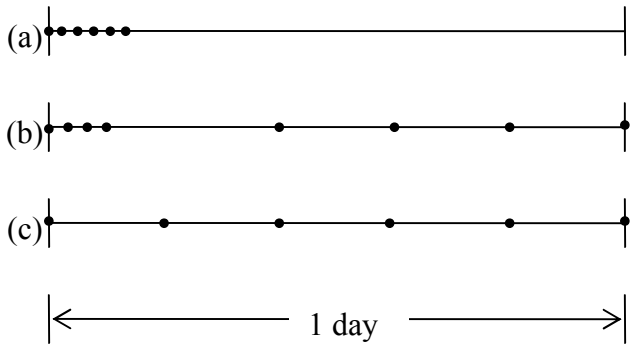
The study presented a formal framework to study the synchronization problem; it defined the freshness and age metrics, which will be described later in section 2. The study described that these policies can be applied on other contexts. The study showed how pages change over time, and modeled as following:

- Uniform change-frequency model: assumed all pages change at the same frequency, this model is useful when:
  - We don't know how often the pages change over time.
  - Pages change at slightly different frequencies.
- Non-uniform change-frequency model: assumed pages change at different rates.

The study described 3 dimensions of the synchronization process:

1. Synchronization frequency: to decide how frequently the whole local copies are synchronized.
2. Resource allocation: to decide how frequently individual page is synchronized, the study classified the synchronization policies into 2 policies:
   a. Uniform allocation policy: to synchronize all pages in the same rate.
   b. Non-uniform allocation policy: to synchronize pages at different rate.

3. Synchronization order: the study presented 3 order policies to solve the problem of ordering the pages to be synchronized:
    a. Fixed order: synchronizing the pages in the same order every time.
    b. Random order: synchronizing the pages in different order every time.
    c. Purely random: user of pages is engaged in ordering the pages by monitoring the selection of the pages.
4. Synchronization points: to distribute the synchronization process over the time of the day for each web site, taking into consideration that in night-time web site is lightly accessed. 3 options to solve this problem were presented, the options are described in figure (1.5).
    a. Synchronize in the beginning of the day, Ex: at midnight.
    b. Synchronize most pages in the beginning of the day, but synchronize the other pages during the rest of the day.
    c. Synchronize the pages uniformly over the day.
    d.



- : Synchronization point

Figure (1.5): Several options for the synchronization points

The study suffered from the following weakness points:

1. Pages history tracking requires more storage space and processing overload.
2. Page change rate suffers from variations, so no change rate calculated using this policy is guaranteed.

The fact of reflecting real change rates using this policy should be considered, history tracking of each page determines the real change rate. This benefit will be used in creating the proposed policy in this study.

1.3.3 Junghoo Cho, Alexandros Ntoulas, Effecive Change Detection using Sampling, 2002

Due to the limited download resources available when web crawler aims to copy the changed pages and to avoid copying the non changed ones sampling policy has been proposed. Sampling idea based on sampling a small number of pages from each web site, then to check these pages on the next download cycle to count the changed ones, then allocating the download resources to each web site according to sampling results, this mean web site with biggest count of changed pages in its sample will get more downloads by the crawler.

This study showed the disadvantages of the change frequency-based policy and compared its results to the change frequency-based policy results based on the ChangeRatio metric as an evaluation metric of the results, the study proposed two main sampling policies:

- Proportional policy
- Greedy policy

Other policies were proposed to suit other crawling cases and to adapt available resources like the adaptive policy. This study showed that the sampling policies perform better than any other change detection policy; these policies will be described in details later in this thesis.

1.3.4 Qingzhao Tan, Ziming Zhuang, Prasenjit Mitra, C. Lee Giles, A Clustering-Based Sampling Approach for Refreshing Search Engine's Database, 2007

This Study [18] casts the problem of finding web pages that have similar change patterns into a clustering problem, it proposed a policy that clusters all web pages from different web sites according to its change rate, and then web crawler samples pages from each cluster to determine its change rate. The study show the features of different types of pages which helps in classifying the web page which also leads to determine it is change rate. The study used a collection of real web pages from the WebArchive project; a special spider was implemented to crawl pages from the Internet archive.

The policy in this study showed better results than the previous policies since it avoid web pages according to their parent web sites, but implementation of this policy requires more overload processing due to efforts used to know the change rate of each

web site whereas the sampling policies group web pages in web sites as they belong to, and it uses lower overload processing when determining change rate for each web site. As described in the change frequency-based policy, it seems the not applicable one due to the required overloading processing and storage needs.

1.4 Contributions

This thesis will propose new approach to achieve better results in detecting the change rate of web pages, which will lead search engines to perform better by providing better search queries results due to its identity to original sources. The proposed approach leads to low resources usage that will lead to minimize the costs of running search engines.

1.5 Outline

This thesis consists of 5 chapters, chapter 1 and 2 explain the freshness issue and the sampling policies, chapter 3 describes the new proposed policy based on the sampling policies, chapter 4 contains the experiments of the study and the results, and chapter 5 contains the conclusions and future work.

Chapter 2

Sampling Policies

In this chapter, a brief introduction about search engines will be introduced, with a description of the sampling principle in change detection, and the metrics used to measure the effectiveness of sampling policy types. These policies will be compared to each other, the adaptive sample size will be studied and described, and solutions for different crawling cases using these policies will be presented.

2.1 Search Engines

Ease of creating attractive web sites using more advanced web technologies brought more users to the internet, they got attracted to the internet to find information most of times and to add it other times. Web contains huge number of web sites now, billions of pages and a huge amount of information that no such library can hold. Some organizations and companies aim to develop better ways to find information on the web in the easiest way. One of these implementations is the search engine.

Search engines work can be summarized in two main categories:

1) Taking a copy of web pages which is done by a part of the search engine called the web crawler.
2) Classifying the web page copy, by indexing it to be reached easily and quickly, and assigning it to search engine queries to appear in the results.

Web crawler is the most complex part in search engine. As mentioned before it is responsible of taking copies of pages and storing them in storage space, reason of complexity is the matter of indexing these pages, how to visit the pages, which pages to visit. A PhD thesis [5] illustrated in Figure (2.1) a high-level architecture of a standard Web crawler.

Web crawler should follow some algorithms and procedures; and consider some issues when visiting a web site because the large amount of data to be processed. Following is a description of these issues.

When web crawler starts crawling a single web site, it prepares the pages to be visited in a list stored in the scheduler to pick an address of the page and take a copy of it.

The order of the pages in that list is important [16] since the web crawler is not going to crawl all the pages in the web site, and only the pages that contain the most important information to be added to search engine storage should be crawled. The order of the pages can be changed during the crawling process when finding new pages which are more important than the current ones residing in the list according to some metrics like the path length between the page and the index page in the web site. Hence web pages come in levels, when trying to add the page to crawling list where level 1 contains the pages that can be reached directly from the index page if it contains links to those pages, and level 2 contains the pages that have links to them in level 1 pages. Web crawlers mainly follow two strategies to fill the crawling list; one of them is filling the list with pages level by level starting from level 1. This is called breadth-first strategy [16]; the other method is filling the list with pages by taking a page from level 1 and it's child pages then move to the next page in level 1 and so on, this is called depth-first strategy. No web crawler can crawl the entire web since web is huge, web crawler should select the most important pages from each web site. This selection depends on the order of pages prepared to be crawled as described earlier like page rank and the importance of information in the page.

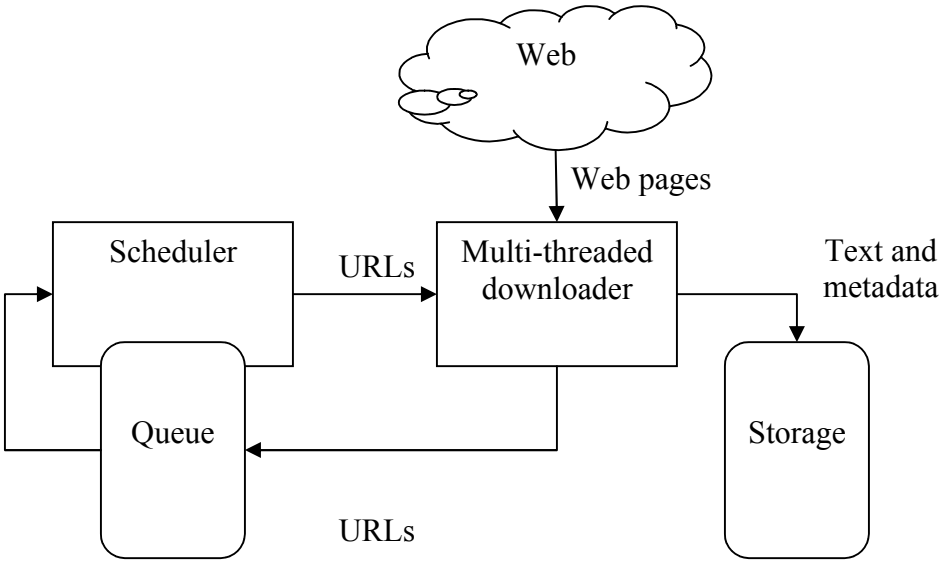


Figure (2.1): High-level architecture of a standard Web crawler.

Determining the importance of the crawled page or pages which are going to be crawled depends on many factors, degree of importance given to the pages is called the page rank [1], that helps in determining the feasibility of visiting the page again looking for new information in it or determining its position in search results when searching in some information residing in it. The importance of information in the page for the people

who search in search engine storage leads to calculate the page rank, many other metrics help in determining the page rank of the page like the length of the path between the index page and ranked page, search engines use different factors to calculate the page rank so that we see different search results in different search engines, the following equation is generally used when calculating page rank of any page:

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)}$$

i.e. the PageRank value for a page *u* is dependent on the PageRank values, for each page *v* out of the set B_u (this set contains all pages linking to page *u*), divided by the number *L(v)* of links from page *v*.

Web server's performance is adjusted to suit human behavior when browsing a web site, but web crawlers have the ability to act as thousands of human users browsing the web site since they can send many requests in parallel to crawl pages in multi-threaded mode. This can affect the server performance if no politeness policy [7] was followed, mainly the politeness policy requires the web crawler to delay between requests to web sites.

2.2 Change Detection using Sampling

Web crawlers should revisit the pages crawled before to check for changes, due to the large amount of pages every crawler maintain this became an issue to solve. Two methods found to solve this issue [6]:

- Uniform revisit policy: visiting all set of pages again, ignoring the rate of change for each page; implementation of this policy is difficult since search engine resources are low like bandwidth and available time, this policy sometimes called the round robin policy [19].
- Proportional revisit policy: visiting the pages more often with higher rate change and discard other pages.

Sampling policies are a proportional policy because they aim to visit the pages that contain changes; three types of sampling policies can be used with preferring the greedy policy for showing effective results under using in real web.

2.3 Why sampling

When assuming web servers would not help search engines in determining the change rate of its pages, like crawling pages with no Meta-tags, change frequency-based policy and sampling-based policies are the most effective proportional policies. The change frequency-based policy clearly suffers from the following weakness points:

- Requires long history of change for every page to estimate the change rate, sometimes change rate changes during history recording which leads to start a new history records and provide unreliable information about the page.
- Recording the history of every page might require huge storage and maintenance overload.

The sampling policies do not require any change history of any page because decisions are based on samples of pages. Random samples taken every crawling cycle emphasize the fairness of these policies with all web sites and all pages in each web site.

2.4 Measuring the Effectiveness of the Sampling-Policies

2.4.1 Introduction

To measure the effectiveness of the proposed policy some metrics should be used in the experiments, regarding the change rate detection of web sites there are three metrics will be described below after that, the selected metric for measuring experiments results and the reason of this selection will be described.

2.4.2 ChangeRatio

ChangeRatio metric is the number of changed pages over the total number of downloaded pages. For example if we crawled 1000 pages from a web sites and detected 300 changed page, then the ChangeRatio for that web site is 0.3, due to the variation in ChangeRatio we take its average over crawling times.

In some cases some pages may have different importance, the weight w_i can be used in ChangeRatio definition to each page p_i where R is a set of pages:

$$ChangeRatio = \sum_{i \in R} w_i . p_i$$

In addition to its ease of implementation of this metric, ChangeRatio can reflect the real change rate of the web site clearly, it is also related to the next metric, and hence it will be used as measurement metric for this study and to compare results.

2.4.3 Freshness and Age

In each web site pages copy stored in the search engine a number of pages are up-to-date at a specified time this number may reach zero, the fraction of this number is the freshness. Freshness of page p_i at time t is defined as

$$F(p_i;t) = \begin{cases} 1 & if\ \ p_i\ is\ up-to-date\ at\ time\ t \\ 0 & otherwise \end{cases}$$

Up-to-date means that page copy is the same as the original one in the web site, the freshness of the entire copied pages (U) in the search engine at time t is

$$F(U;t) = \frac{1}{|U|}\sum_{pi \in U} F(pi;t)$$

Age metric represents the time a page spent in the search engine with out re-indexing, means asking: how old the page is? Age is defined as follows:

$$A(pi;t) = \begin{cases} 0 & if\ p_i\ \text{is up - to - date at time} t \\ t - \text{modification time of } p_i & \text{otherwise} \end{cases}$$

And age of the entire pages in the search engine is

$$A(U;t) = \frac{1}{|U|}\sum_{pi \in U} A(pi;t)$$

Age and freshness metrics are hard to measure so that using an easier metric is preferred. Age metric shows the age of the page, sometimes page age should remain old if the page change rate is very low, some finding pages with high age is not evidence of low performance.

2.4.4 Divergence

This is a divergence value represents how different the page copy from the original one, this metric is very general and un-usable.

2.5 Main Sampling Policies

2.5.1 Introduction

As the sampling has been described idea earlier, previous studies [6] defined two main sampling policies and other two policies derived from them suit different cases like adapting sample size and crawling under low download resources. The two main policies were classified according to the used way in change detection and being fair in sampling all web sites.

2.5.2 Proportional Policy

This policy distributes download (crawling) resources over web sites according to change rate of each web site calculated using samples. This policy relies on sampling to detect change rate of each web site, this policy can be developed to use a fixed sample size or changed sample size, but before getting involved in this, effectiveness of the policy should be determined and compared to other sampling policies, and through an example will demonstrate that other sampling policies reflect better results in change detection. This fact will be demonstrated with some strange other facts like other policies are not fair in crawling all web sites. This means some web sites may not get enough crawling.

This policy maintain each web site as an important effective part in improving the performance of the policy but it may ignore the web site if no change is detected in its sample. This policy samples pages from each web site it maintain, calculate change rate of each web site according to change detected in the sample, then calculate each web site quota of download resources so web sites with higher change rate got large download resources. Download resources means number of pages crawled from each web site. The following equation can be used to calculate this quota of download resources:

$$Q = \frac{\frac{c}{z}.D}{\sum_{w \in S} \frac{c}{z}}$$

Symbol	Description
Q	Quota of download resources
c	Number of changed pages in each sample
z	Sample size
D	Download resources
w	Web site
S	Set of web sites

Table (2.1): Notations used in proportional policy equation

2.5.3 Greedy Policy

This policy consumes all download resources on web sites with high change rate only. Web site change rate here is also calculated using sampling. After reviewing change rate for all web sites, a threshold is calculated to determine which web sites will be downloaded and which ones will be discarded from download. In other words, the greedy policy [10] prepares a list of selected web sites to be downloaded, web site with higher change rate resides in top of the list, and other web sites are ordered in the list this way. Downloading pages in this policy is different because when download starts web crawler in this case should download all pages from the web site with higher change rate until complete downloading all pages, then the crawler moves to the next web site in the list if more download resources are available and keep consuming the download resources by downloading pages until finishing the download resources.

We notice that some web sites are not going to be crawled since they have low change rate and this is the greedy principle that aims to promote web sites with high change rate and ignore others because they suffer from low change rate, download parameter that determine wither to download a web site w or not according to its change rate c_w where T is change rate threshold that can be used to compare change rate of each web site with it to decide wither to download pages from the web site or not, is defined as:

$$F(c_w) = \begin{cases} 1 & c_w \geq T \\ 0 & c_w \langle T \end{cases}$$

This policy seems to be unfair with some web sites because they may not get downloaded at all if they provide low change rate, we might think it is not effective and does not reflect good results of detecting change rate, but with some calculations the

greedy policy appear to be the best sampling policy. This fact is true because greedy policy directs all its download resources to the group of pages that contain the most changes and ignores others. It considers the web sites one group rather than considering each web site an effective part of reflecting change detection results. The next section contains an example that describes the two policies better and proves the effectiveness of the greedy policy. More development of it will create new policies that fit other cases. Also a new policy based on the greedy policy will be proposed in this study by combining it with the change frequency-based policy.

2.5.4 Greedy Policy versus Proportional Policy

The following example shows how greedy policy can detect changes better than the proportional policy; the example also helps in describing the two policies. Example: Assume we got two web sites, each one contains a 100 pages, and we can download 100 pages, used 20 of them to sample the two web sites, in the first web site, we detected 7 of 10 pages has been changed, and 4 pages in the second one. According to the proportional policy the rest of pages can be download which are 80 will be divided and used to crawl part of the 2 web sites, for sure the first web site will get the biggest amount of downloads according to its sampling result, number of page downloads for the first site will be $80 \cdot \frac{7}{7+4} = 51$, and for the next web site will be $80 \cdot \frac{4}{7+4} = 29$. But in greedy policy the 80 page downloads will be used to download the first web site.
We might think the proportional policy is better because it is fair with the two web sites, but according to some calculations the greedy policy showed better performance in change detection. When using the 80 page downloads to download pages from the first web site we will detect $80 \cdot \frac{7}{10} = 56$ changed pages according to the sample result, but in the proportional policy $(51 \cdot \frac{7}{10} = 36) + (29 \cdot \frac{4}{10} = 12) = 48$ changed pages will be detected. Hence greedy policy performs better than proportional policy; any new policy should be based on the greedy policy to achieve the best freshness problem solving.

2.6 Sample Size

The study [6] that presented the sampling policies showed how pages assigned for sampling can affect the crawler decision of selecting the pages and reflect different results. For sure small sample size will reflect bad download decision of web sites and will make the sampling policies perform as the Uniform policy (Round-robin) where

search engine crawl all pages in each web site and takes longer time till visiting the page again for taking another copy, and this may miss changes added to the page. The sampling policy will perform this way because the small number of page downloads assigned for sampling, while using the other pages in crawling web sites due to the weak decision taken by the crawler to select the pages for download which leads to crawl in the same situation every time so that stability is seen in change rate of the greedy and proportional policies. Also big sample size requires decreasing the number of page downloads and uses it in sampling the web sites; it also leads to perform as the Uniform policy since most of page downloads is used in sampling and small number of them is used in downloading pages, so web crawler will select random pages for sampling every time and they will reflect the same change rate. Figure (2.2) contains a chart that describes how sample size affects the performance of the used policy as suggested in the study [6] that described this case.

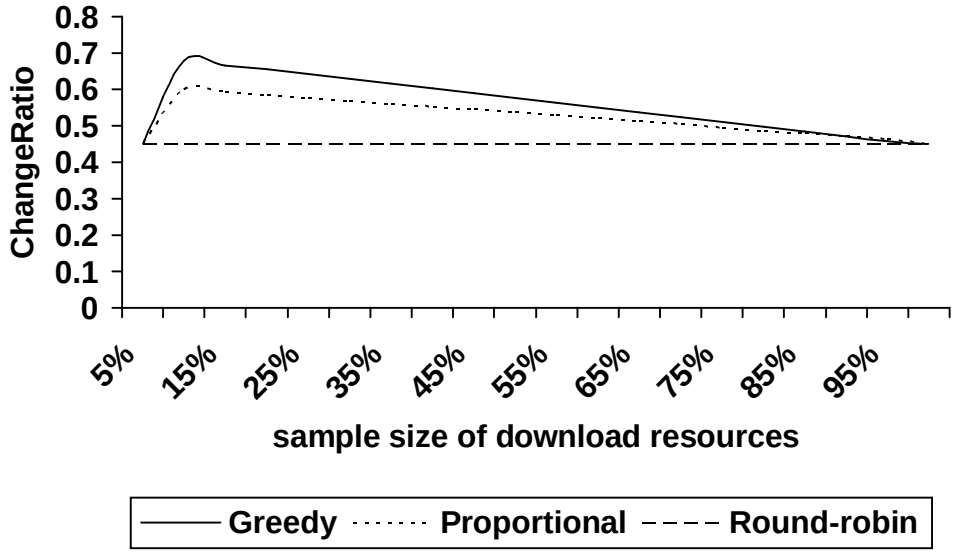


Figure (2.2): Expected ChangeRatio for various sample sizes

Figure (2.2) shows same performance for Greedy, proportional and Round-robin policies when sample size is small and when it is large, but high change ratio is seen for the two sampling policies when number sampling pages is low but not near to zero.

As development of the greedy policy to perform better will be done, greedy policy should not use a constant number of sample size every time. Sample size should adapt with the change rate of each web site. An equation proposed in previous study [6] to calculate the sample size in each download cycle according to different metrics related to the download process, following is the used equation and set of notations used:

$$S = \sqrt{\frac{Nrf(p_t)}{6(\overline{p}_r - \overline{p})}}$$

Notation	Description
s	Sample Size
N	Average number of pages in all sites
r	Ratio of download resources to the total number of pages we maintain
f(p)	Density function of all web sites
p_t	Threshold p value. If $p_i > p_t$ for some S_i, then S_i belongs to the highest $100.r\%$ sites
p_r	Average p value of web sites having $p_i > pt$
p	Average p values over all web sites

Table (2.2): Notations used in sample size equation

This equation can be used when we know p value for each web site; p is the fraction of the number of changed pages to the total number of pages in each web site. Since greedy policy consumes the download resources on the web sites got the larger number of the changed pages, some web sites may not got crawled because value of p didn't reach the threshold of p value which defined as p_t in the equation.

2.7 Adaptive Sampling

Another implementation of the greedy policy was proposed [6] by detecting change rate in early stages of sampling and deciding whether it is useful to complete downloading from the web site or not, the new policy called adaptive policy because it adapts while sampling, according to the change rate of the current web site being crawled. The algorithm in figure (2.3) describes how this policy works; In this algorithm the site is sampled early, then p value is calculated and compared to the threshold that qualify the web site to be crawled or not. This way will save a lot of download resources and let using them in crawling other pages, which will lead to direct download resources to other pages with high change rate as the greedy policy.

2.8 Low Download Resources

In case we are crawling pages under low download resources, we might sample small number of pages from each web site which does not reflect the real change rate of the

web site, as described above; low sample size will make the sampling policy act like the uniform download policy.

Algorithm: Adaptive-sampling policy
Parameters:
α: confidence level (a value between 0 and 1)
κ: number of pages to sample in each iteration

Procedure
[1] S = {S_1, S_2, ,,,, S_n} // Set of sites to be sampled
[2] Loop while I have download resources
[3] For each site $S_i \in S$
[4] Sample κ pages from S_i
[5] p_i = Estimate of p value for S_i base on the samples so far
[6] (l_i,h_i)= 100. α% confidence interval for p_i
[7] Compute threshold p_t from the distribution of estimated p_i's
[8] For eachWeb site S_i in S
[9] If ($h_i < p_t$) S = S - S_i
// p_i too low. I do not download from S_i
[10] If ($p_t < l_i$) download all pages in S_i and S = S - S_i
// p_i very high. I download pages from S_i

Figure (2.3): Algorithm of the adaptive-sampling policy

Previous study [6] proposed a solution for this scenario by combining any of the sampling policies with the uniform download policy. In other words, the solution proposes grouping web sites into number of sets and using the uniform policy in the crawler to pick a group of web sites in each cycle and implement one of the sampling policies on it and download.

This solution can be described in the following example. Example: A search engine maintains 250,000 web sites and we have download resources of 1,000,000 page download, means we can download 1,000,000 page only every cycle, and our cycle duration is 1 week, means we download that number of pages every week, without following the proposed solution each the crawler will crawl 4 pages from each web site if we assume each got 100 pages to download, this way we will miss many changes occurred in pages content, if we follow the proposed solution; we have to group every 10,000 web site in a group and in each cycle we pick one group and implement the sampling policy and this might give better download chances for all web sites but will

lead to miss changes sometimes for long times, figure (2.4) describes this approach for downloading under low resources.

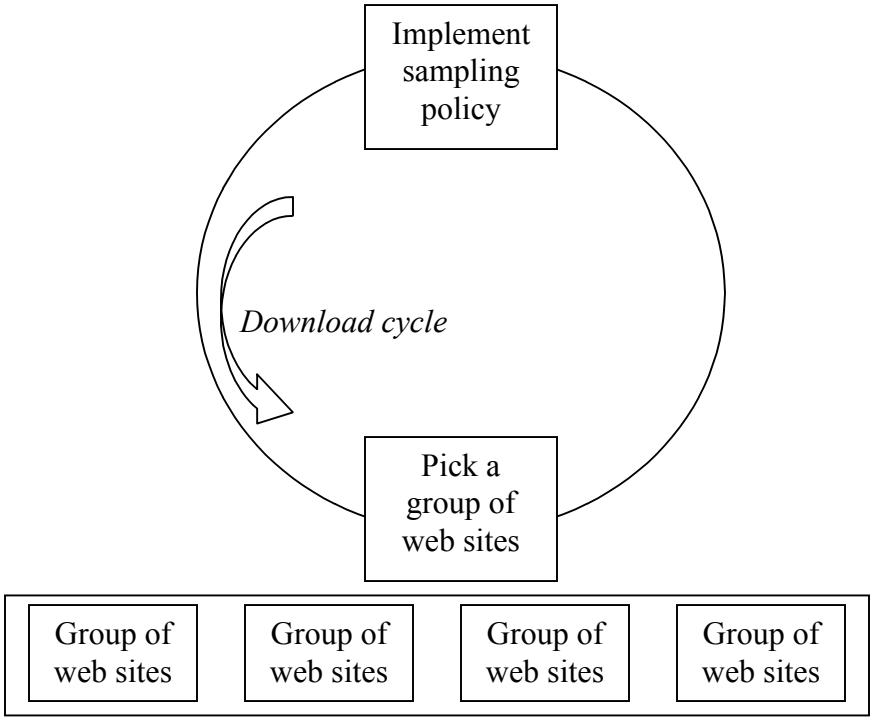


Figure (2.4): Downloading pages under low resources

Chapter 3

The Proposed Policy

In this chapter, some important parts in greedy sampling policy and change frequency-based policy will be described in details, new policy created by the combination of the two policies will be proposed, analysis the requirements for designing it will be performed, with showing the proposed policy hierarchy and how it works.

In the previous chapter, the fact of best change detection results is done by the greedy policy has been proved, and modifications added to it to handle different crawling cases proved it is sufficiency to web crawlers. The description added in chapter 1 that talks about the study that implements the change frequency-based policy shows that this policy also reflects real change detection results better than the sampling policies because it detects the real change rate of each page using its history with excepting the issue of load and resources needed to implement this policy. This thesis will provide a policy that uses the success points in the two policies to achieve proposing new policy, the idea of the policy is to consider the greedy sampling policy the base of developing the proposed policy but when it reaches change detection part in the sample the change frequency-based policy will be used to calculate change rate, to reflect the real change rate of the sample which will lead to the better results and uses accepted level of load and resources.

3.1 Change Frequency-Based and Greedy Sampling Policies

Although the change frequency-based policy [8] is overload implemented due to page change history monitoring, requires large storage, and change rate variation for each page, this policy can be used to reflect good change detection results by combination with the greedy-sampling policy. It can be used without tracking change history for all pages but to track it for pages used in the sample when the greedy policy pick a random pages for sample, this way tracking change history for few pages is needed only, using smaller storage for page monitoring since page will be discarded from the sample on the next download cycle when picking random pages, and this leads to delete its old versions and earn reserved storage by the page, regarding the change rate variation it is more efficient to track the history of the page for few download cycles so no big variation in change rate is detected and low overload on the crawler, also monitoring the change in few pages

does not show change rate variation too much. Figure (3.1) shows how the new policy works.

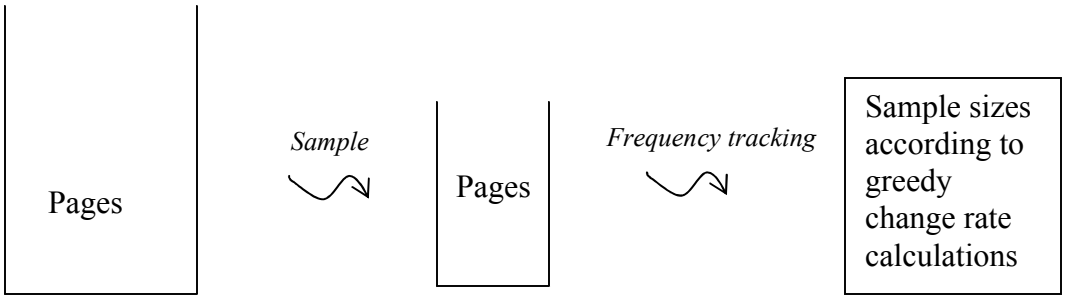


Figure (3.1): Illustration of the proposed policy

3.2 Requirements Analysis

The proposed policy should contain a number of old versions of each page used in sample; number of page versions to be tracked should be flexible to adapt the variation of page change rate. Good change detection technique should be used to reflect real occurrence of change in page history tracking.

As the goal is to combine two policies, the main consideration is the heavy load of the implementation. Most of the load might occur by the change frequency-based policy part in the new policy since pages should enter a process of history tracking. The question of how many versions of the page should be tracked to detect its change rate should be answered, when designing the policy, the possibility of variety in the change rate for the page is low in this few versions of it, tracking more versions of the page seems good but the possibility of getting wrong results of change rate is higher because this long history tracking may include higher possibility of change rate variation, and most of times few pages tracking reflects change rate values near to the real ones. Agility of the greedy-sampling policy should be considered because it deals with a small number of pages taken from the sample, so that this policy plays the biggest part in the proposed policy.

As the goal is to keep old copies of pages used in samples, the storage and indexing issues of these pages should be considered, and find a technique to decrease the space used for page's history saving because any normal used web crawler deals with huge amount of pages which already reserve large amount of space, the issue of allowance of deciding the number of copies to track should be solved, because more copies of pages require more storage space. Indexing the copies of each page is important factor in refining the performance of the new policy, because fast page reaching saves time.

The proposed policy should select random pages for creating each web site sample every time, and each page should have the specified number of old versions when it participate in change rate calculations. As mentioned above if each page in samples contain the specified number of old versions, all versions should participate in calculating the change average of the page over its history. This part of policy implementation will be the change frequency-based policy usage and the other parts of work will be in the side of using the greedy sampling policy.

After determining change rate for each web site number of pages reserved for sampling should be distributed over the web sites according to change rate in a greedy methodology where web site with high change rate gets more pages for sampling and low change rate web sites got few pages. Due to the wrong results that might appear by samples which do not reflect real change rate of the web sites, some sampling pages should be reserved for web sites with zero change rate to give them the chance of providing changes in the small given sampling pages.

3.3 Policy Design

3.3.1 First Sample Size

Since distribution sampling pages over web sites is not possible when each web site change rate is not determined in the beginning of policy implementation a size of first sample should be determined, this size should be used will all web sites used in this policy, and this simple equation is used to calculate the size:

$$F = \frac{P}{W}$$

Where F is the first sample size, P is the number of pages used in sampling and W is the number of web sites used.

3.3.2 Page History Tracking

3.3.2.1 Flexible Number of Page Versions

Number of page versions used in sampling has been made changeable due the variation of change rate, high variation requires small number of old page versions to avoid getting

wrong change rate of the page, if certainty of low variety of change rate is high, bigger number of old page version for tracking is preferred but with consideration of other factors that might affect the performance which are described in section 1.3.2.

Using a reasonable small number of page versions in change detection for each page at the beginning is recommended because it satisfies the requirements of avoiding detecting big variation in page change rate and to perform low load when implementing the policy, when considering the fact of load occurrence when tracking history of each page. In some other cases the policy might be used with no consideration of heavy load implementation and confidence of small variation in pages change rate over its history. Figure (3.2) shows a flow chart which describes the design of the part responsible of page history tracking for more information about how was this part of the policy designed see code in appendix A.

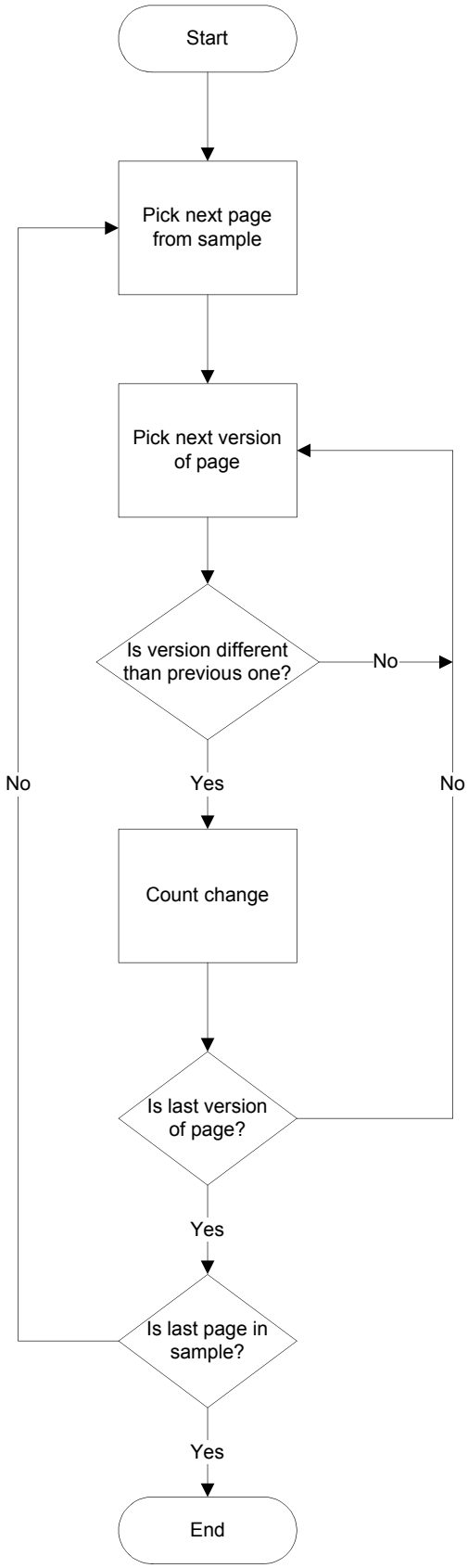


Figure (3.2): Page change rate calculation

3.3.2.2 Page Versions Storage and Indexing

As required to limit storage space used due the large amount of pages maintained, the following equation is used to determine the percentage of needed storage space to the total number of web pages:

$$S = \frac{V \cdot P}{T} \cdot 100\%$$

Where S is the needed space percentage of the original used space, V is the number of page version for history tracking, P is the number of pages used for sampling, and T is the total number of pages used.

Recording dates of change occurrences in each page used to track one page history only is recommended, rather than storing the whole old copy of the page. This way any extra space for history tracking is not reserved and changes is monitored over time of page history collecting, not in the same moment of change average calculation for the page, means this value can be calculated by counting the number of different change dates and dividing it over the number of page old versions used in history tracking, this can be defined as

$$C(w) = \frac{D}{V}$$

Where $C(w)$ is the function that calculate the average of change of the page over its history, D is the number of different dates recorded over pages history, and V is the number of page versions used. Indexing these changes becomes easiest when using the way of recording dates of changes, databases can be used to reflect fast data reaching.

3.3.2.3 Change Detection

In tracking the history of each page, change should be monitored in old versions of the page, each version of the page should be compared to the previous version of it and counting the change if found, after running this process on all versions of the page changes should be counted and average of it should be determined over the number of the versions. This process that contains the change frequency-based policy part in the proposed policy, contains the mentioned load that should be reduced. It also requires storage space since many copies of each page used in sampling are stored, and this storage space should be reduced also, in the new policy design the way to solve this is to

use another approach to monitor change when comparing the versions of the page, this way should do this comparison every time new version of the page is added, to distribute the load over time of versions collecting, and this should be done by recording the date of the change in each comparison and discard the older page after comparison to save storage space and to deal with dates rather than dealing with pages stored as files which will require more processing, discarding the older version of the page will let us handling one version only of the page used in the sample, means handling one extra file for each page, and this can be defined in the equation used in section *3.3.2.2* after modifying it to look like this:

$$S = \frac{2P}{T} \cdot 100\%$$

Here we duplicated the number of P which show the number of pages used for sampling, and S is the needed space percentage of the original used space, and T is the total number of pages used.

3.3.3 Greedy sampling Implementation

As sampling pages are going to be distributed over web sites, the following equation is used to calculate each web site number of sampling pages:

$$F(w) = \frac{c_w \cdot T}{\sum_{w \in S} c_w}$$

Notation	Description
$F(w)$	Function that returns the sample size for web site w
c_w	Change rate of web site w
T	Total number of pages used for sampling
S	Set of web sites

Table (3.1): Notations used in sample size calculations for the greedy policy

Samples are not 100% accurate, and results of zero change rate web sites are not certain in all times, and these web sites should get another chance to provide changes in the next crawling cycle, so the goal is to reserve small number of sampling pages for these web sites, hoping to find missed change in them.

3.4 Policy Implementation

The parts of the policy created in the design phase has been aggregated and implemented to work as described in figure (3.2) where we can see the cycle of crawling keep running, below there is a loop of web sites which contains the process of sampling web site and calculating change rate which described in details in figure (3.3), we notice that sample size calculation is not in the same loop because sample size for each web site is calculated with considering other change rates of web sites.

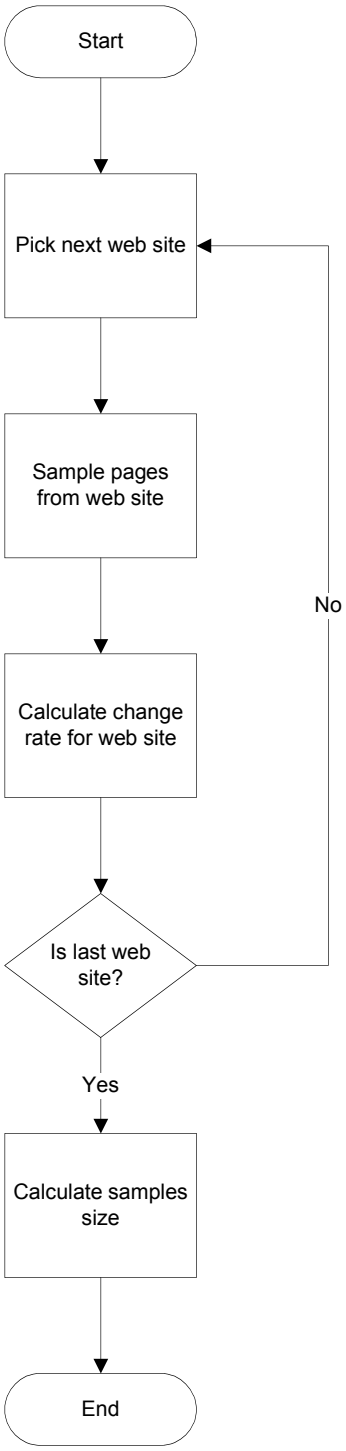


Figure (3.3): The proposed policy crawling cycle

Chapter 4

Experiments

This chapter contains a description of tools used to crawl and monitor web sites that leaded to create the web sample; it will contain a description of the tool developed to perform some specified functions for this thesis, like some facts about the used web sample, and implementing the greedy policy and the proposed policy. Implementation results of the policies will be described

4.1 Used Tools

Web sites crawling needs a program to act like the web crawler to discover links and download pages and read added in each page, any program should provide an effective way to detect change in pages, and accept changes and modifications to customize the web crawler to suit effective implementations. Every web crawler picks web sites from prepared list; it downloads the first page returned by the server, web crawler stores the page and the needed information of it, and then scans it for links to other pages, and other pages will be added to the scheduler to prepare them for crawling in the same way used in first page. Web crawler should find a way to stop crawling one web site if it gets into links loop where pages contain links to each other and web crawler will keep following this loop of tracking pages found as link in each crawled page; Sometimes the web server provides infinite links generated to produce dynamic links, like dates links where a web developer generates a calendar in the web for all the possible years a computer can handle; Each day contains a link to display its events, and this dynamic calendar web may contain infinite links. This issue should be handled and finished by the web crawler. Web crawler should allow selecting the depth of links crawled which may lead to reduce the unnecessary pages and to avoid getting into infinite links loop.

An open-source web crawler [17] called Netcrawler has been used and modified to crawl the list of pages prepared for the sample. This web crawler is developed using C# language and uses sockets to crawl pages which is better than using HTTP requests in C# because it provides us with the HTTP header coming with page when crawling it. The HTTP header contains information about the page stored in fields, like (Last-modified)

field that tells when the page was modified and helps in detecting the change date of the page, now a day's few web sites add this field in the HTTP header because the used web servers which do not support this feature, so that few options of web sites were available to be chosen for the sample. Other web sites add this field to its HTTP header, but with fixed value most of the times; this value is the date of requesting the page. The reason of this is also the used technology in the web server that builds dynamic generated pages, means the used page is written once then the web server uses it as a template to fill it with data retrieved from database, sometimes determined according to parameters added to the page when requesting it, Example: Assume an HTML page written in PHP language [12] with the name (*topic.php*) is used to display topics retrieved as text from MySQL database [14], parameters are used to identify each topic and generate links to the topics. The parameter (*topic_id*) will be used as unique identifier to topics, so link to one specified topic will look like this (*topic.php?topic_id=3*).

Another tool has been developed for this study. It calculates real change rates, and implements policies for generating performance results and comparison; the tool is developed to support the following points:

- Database clean and maintenance from useless rows or damages.
- Calculating real change rate of each web site.
- Implementation of the Greedy-sampling policy.
- Implementation of the proposed policy.

C# .Net [2] language has been used to develop the tool, MS Access database has been used to store results. The tool has not been developed to crawl all pages in web sites, since the goal is to crawl the same pages every time, and to calculate sample sizes that fit the change in each web site and to determine the date of change occurrence. The tool can be developed to manage crawling process according to calculated sample sizes when implementing the policy, in that case the suggested program should crawl all pages found in all web sites if it satisfies crawling options when considering other crawling issues like depth of the page, program should deal with new links found and new changes of web sites status.

4.2 Web Sample

Experiments are performed over a web sample contained more than 1000 pages collected from 9 web sites; web sites vary in change rate and pages number. The same pages have been crawled every day for a duration of 3 weeks, new page links found every time were

ignored because the goal is to have a sample that contains pages crawled for a complete 3 weeks. Sample contained this selection of web sites to improve freshness in web sites that belong to local region of Jordan. Most of the web sites visitors are coming from Jordan; the reason of the selection of these web sites is it is convenient to the method used for change detection.

Total number of pages used for the sample is 1140. There was a problem of choosing the web sites that do not include the Last-modified field in its header. The issue of including different web sites characteristics in the experiments sample was considered to implement the experiments in environment similar to the real web, so web sites should vary in change rate and number of pages. The selection of the pages performed after crawling pages from the nine web sites on depth of level 2, which leaded to create a list of 1140 pages; These pages vary in change rate and convenient to the used change detection technique. The difference of web sites content in the sample was considered for the same reason of getting a sample similar to real web, means different types of web sites, as seen in the web sites names, two web sites have educational domain name (.edu), they belong to two universities in Jordan. These web sites also differ in change rate, experiments sample should also contain news web sites since this type of web sites got many useful characteristics like different page types with different change rate, size and content, 3 news websites has been selected for the experiments, these web sites also differ in change rate, sample should contain other different types of web sites like commercial and governmental ones, two governmental web sites suit the way used in change detection has been found, its domain name ends with (.gov); They vary in change rate also, other web sites that have very low change rate have been added because web contains many websites suffer from low change rate and many of them have been added to the web to publish some information that does not need any changes.

The crawler has been modified to store the needed information for the experiments in MS Access database like crawling date, page URL, last-modified field and size of the page. After web sites crawling completed, the percentage of average change rate for each one has been calculated. This reflected the required variety in the sample as shown in table (4.1).

Each change rate value has been calculated by detecting the change in pages after comparing them to the same copy crawled in the day before, after completing crawling duration; the average of change rates for each web site has been taken. Web sites have an advantage of low change rate variety which is very useful in generating the results since detecting change rate variety should be avoided by decreasing the tracking of page history as mentioned in earlier sections. Figure (4.1) shows change rate variety over time for the web sites used in the sample.

Web site	Change rate	Number of page
www.alarabiya.net	3.27%	528
www.alaswaq.net	25.77%	223
www.alzaytoonah.edu.jo	2.22%	38
www.jordandaily.com	81.95%	20
www.jordanislamicbank.com	0%	45
www.madabacity.gov.jo	0%	9
www.moe.gov.jo	1.54%	110
www.mutah.edu.jo	43.59%	94
www.nitc.gov.jo	9.09%	73

Table (4.1): Sample web sites, their real change rate and number of pages

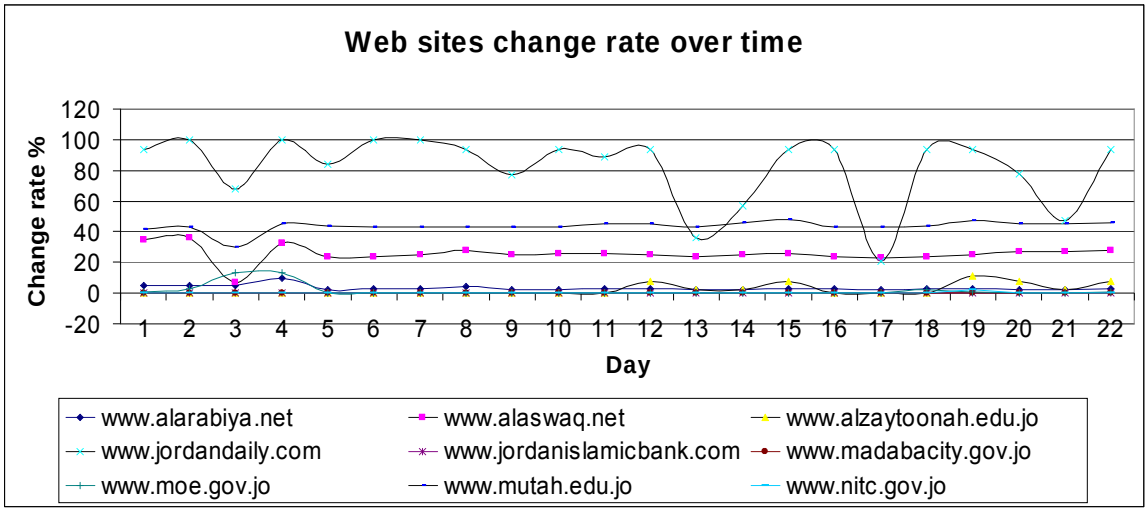


Figure (4.1): Web sites change rate variation over time

4.3 Greedy-Sampling Policy Implementation

In the experiments greedy algorithm was implemented to distribute sampling pages over the web sites, so web sites with high change rate get the biggest amount of pages for sampling and so on, while the web sites with low change rate may not get any page for sampling because they were consumed before, but this way is not fair with some web

sites and they should have a chance to improve having changes, so each web site with zero sample size got very low number of pages for sampling.

As change rate of each web sites is not know in the first time the crawler is going to determine the samples sizes, this study suggests 10% of total maintained pages to be for sampling, which is 100 pages according the sample of 1000 pages and more, as this was implemented in the used sample for experiments each web site got about 10 pages for sampling, in the next cycle same pages in the sample were monitored to count changes and to determine the number of pages for sampling in each web site, and so on until implementing to the specified day of experiments duration, then how this policy was monitored and recorded its results of knowing change rate of each web site. This policy was implemented over different sizes of sampling pages and showed very good results of change-rates near to the real results as shown in table (4.2), where accuracy remained high until using 10 pages for sampling and was very accurate also. Appendix A contains the design of the part that runs the greedy policy on the experiments sample.

4.4 Proposed policy Implementation

High load generated by the proposed policy is expected. For sure this load is higher than the load done by the greedy policy because new change detection way is used, but this way is only used when determining the change rate of a page, better results are expected, despite greedy policy implementation in the previous section showed high accurate results. This policy reflected near results to the Greedy policy with higher accuracy and better advantage of keeping this accuracy higher than the Greedy policy under fewer pages for sampling. The reason of these results is the accurate change rate calculated for each page in each sample according to its history, high load is noticed and was higher than the load monitored in the greedy policy implementation but it is expected. The fact of restructuring the tool used for implementing the policy will perform low load should be considered. Also there will be another technique in history tracking will be used in real implementation which is going to reduce the load; In this implementation case we track page history while we crawl the page for different time, while here in the thesis experiments, tracking all versions of each page is done at one time, as described earlier this the change frequency-based policy part of the proposed policy and it's the one the performs with high load. Table (4.2) shows the records of change-rate detection in each policy and under different cases.

The following graph shows comparison of the greedy policy and the proposed policy over different number of pages used for sampling, results evaluated by accuracy to real change rates of the web sites:

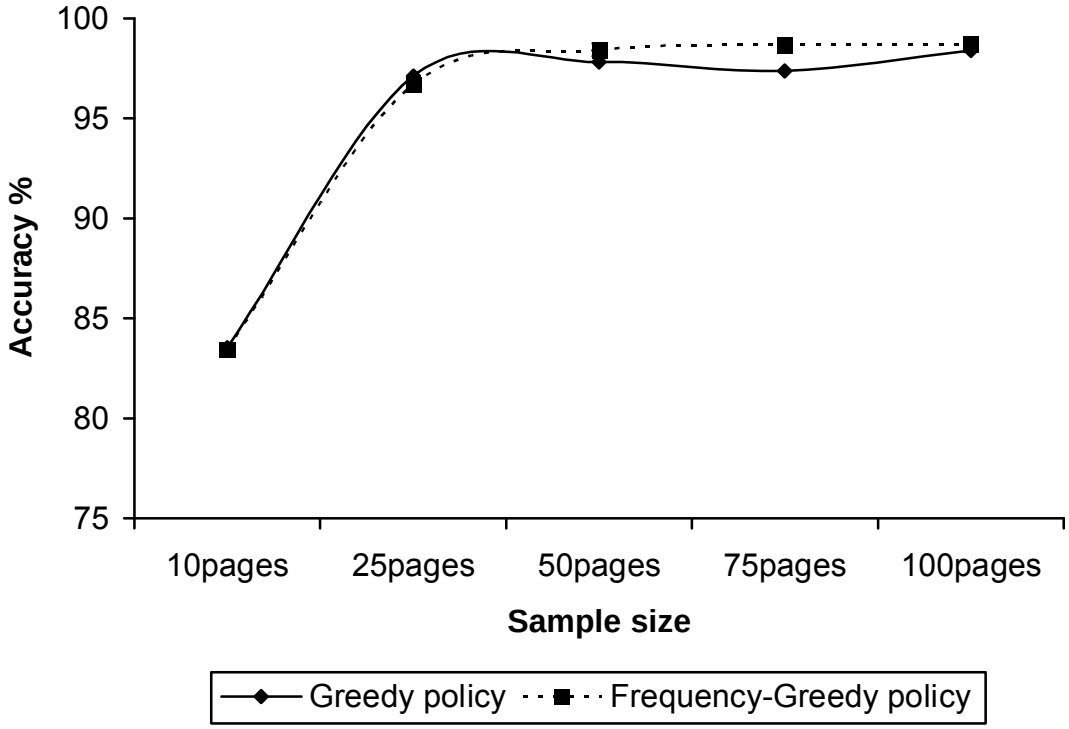


Figure (4.2): Real change rate using the different policies

Table (4.2): Change-rate detection using different policies and cases

	Real	Greedy policy					Frequency-Greedy policy				
Site domain	Real change rate	**Sample:** 10pages	25pages	50pages	75pages	100pages	10pages	25pages	50pages	75pages	100pages
www.alarabiya.net	3.27	0.00	0.57	0.00	1.57	5.19	0.00	0.00	1.20	5.75	4.40
www.alaswaq.net	25.77	7.52	35.19	29.24	16.10	23.19	2.50	36.50	25.70	25.25	23.85
www.alzaytoonah.edu.jo	2.23	0.00	0.00	2.38	0.00	0.00	0.00	0.00	0.60	2.65	2.50
www.jordandaily.com	81.95	11.67	76.43	74.76	75.52	76.24	4.40	78.20	74.60	75.25	75.25
www.jordanislamicbank.com	0.00	6.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
www.madabacity.gov.jo	0.00	6.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
www.moe.gov.jo	1.55	1.19	1.19	0.57	1.57	1.52	0.00	6.25	3.50	0.85	2.70
www.mutah.edu.jo	43.59	2.38	37.90	39.29	40.19	45.52	2.50	38.70	42.55	42.65	43.35
www.nitc.gov.jo	0.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Percentage of accurate by comparsion to real change rate.											
www.alarabiya.net		96.73	97.30	96.73	98.30	98.08	96.73	96.73	97.93	97.52	98.87
www.alaswaq.net		81.75	90.58	96.53	90.32	97.42	76.73	89.27	99.93	99.48	98.08
www.alzaytoonah.edu.jo		97.77	97.77	99.85	97.77	97.77	97.77	97.77	98.37	99.58	99.73
www.jordandaily.com		29.71	94.47	92.81	93.57	94.28	22.45	96.25	92.65	93.30	93.30
www.jordanislamicbank.com		93.67	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
www.madabacity.gov.jo		93.67	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
www.moe.gov.jo		99.65	99.65	99.03	99.97	99.98	98.45	95.30	98.05	99.30	98.85
www.mutah.edu.jo		58.79	94.31	95.69	96.60	98.07	58.91	95.11	98.96	99.06	99.76
www.nitc.gov.jo		99.91	99.91	99.91	99.91	99.91	99.91	99.91	99.91	99.91	99.91
Average		83.52	97.11	97.84	97.38	98.39	83.44	96.70	98.42	98.68	98.72

Chapter 5

Conclusions and Future Work

5.1 Conclusions

As web became more dynamic and new added technologies facilitate changing the content of web pages, copies of pages stored in search engine should be considered old and need to be refreshed when there are available resources to do this. This will lead to get useful search results for users. Web crawler which visits the pages cannot take copies of all pages in the web because they are too many, pages to be crawled should be selected and the web crawler is responsible of selecting these pages with consideration of achieving best crawling quality. So even if the size of the web is limited for the crawler this limited part of the web should contain the pages with the most needed information by the users and to contain the new added changes, so crawler is required to crawl part of the web and to keep crawling this part over time to contain new changes added. The easiest way to re-crawl pages again is to follow the round-robin policy where the crawler cannot crawl all the pages again, each time it re-crawl it picks a part of the pages and crawl them again, then on the next time it picks another part and so on, selection of the pages in each crawling cycle is not determined; It occurs randomly, so many changes will be missed in some pages during crawling cycles. Another crawling policy that selects the pages which contain changes for crawling in each crawling cycle, will perform better because all pages will be identical to the original source, because pages that contain changes has been crawled in crawling cycle.

Detecting the changed pages is complex so many implementations of the search engines preferred to use the round-robin way [6], but when determining the feasibility of using a policy to detect changes search engines may perform better, one solution was to let web servers submit the date of each page change as HTML tag in each page, this solution may not be used by all web servers since they can submit pages to the web without adding the HTML tag despite they will get content submitting to the internet. Another solution suggested monitoring each crawled page changes to know when it is going to change and select it for crawling in the crawling cycle when it is the date of the change, as described in the thesis this solution reflects real results but require too much resources. Sampling pages from each web site to determine its change rate was another solution so web crawler can distribute its efforts over web sites that contain most changes; One implementation of this solution is the greedy sampling policy that directs

the web crawler efforts and downloads to the web sites with high change rates to detect as much as it can of the changes; This policy showed the best results of determining the changes in each web site, and to know when pages are going to change, the policy can be perform more effectively when it determine the sample size for each web site adaptively and determine wither to crawl pages from the web site or not during sampling it, this sampling policies can be implemented under low number of pages for sampling.

Change frequency-based reflects real change rates of web sites; This thesis proposed a policy that uses the change frequency-based policy to detect change rate in determining the change rate of pages used in sampling in the greedy sampling policy, this combination showed better results, and performed better even with low number of pages used in sampling, it also use few download resources for determining change rates near to the resources used in the sampling policies.

5.2 Future work

The area of developing policies based on the greedy policy is interesting to present new policies that can detect very accurate change rates close to real ones, and many other crawling cases can be considered to adapt policies for problem solving under these cases.

Difference between the proposed policy and any other policy like the greedy policy will appear clearly in larger real web sample and longer monitoring period; researchers who handle such amount of data can present contributions with different web samples.

The proposed policy can be developed by software engineers to follow a development life cycle since these steps were not implemented in advance, means more specifying in the new policy requirements and design will result better implementation of it and better accuracy.

The proposed policy should be implemented under different cases, like low download resources. Adaptive sampling policy can be developed to be combined with Change frequency-based policy to produce a new policy and considered as contribution.

References

[1] Andrew Clausen (2004). The Cost of Attack of PageRank, The International Conference on Agents, Web Technologies and Internet Commerce (IAWTIC'2004).

[2] Archer, Tom (2001). Inside C#. Microsoft Press. ISBN 0-7356-1288-9.

[3] Baeza-Yates, R., Castillo, C., Marin, M. and Rodriguez, A. (2005). Crawling a Country: Better Strategies than Breadth-First for Web Page Ordering. In Proceedings of the Industrial and Practical Experience track of the 14th conference on World Wide Web, pages 864–872, Chiba, Japan. ACM Press.

[4] Berners-Lee, Tim; Bray, Tim; Connolly, Dan; Cotton, Paul; Fielding, Roy; Jeckle, Mario; Lilley, Chris; Mendelsohn, Noah; Orchard, David; Walsh, Norman; Williams, Stuart (2004). "Architecture of the World Wide Web, Volume One". Version 20041215. W3C.

[5] Castillo, C. (2004). Effective Web Crawling. PhD thesis, University of Chile.

[6] Cho, J. and Alexandros Ntoulas, (2002). Effective Change Detection Using Sampling. In Proceedings of the International Conference on Very Large Databases (VLDB), Hong Kong, China.

[7] Cho, J. and Brandman, Onn and Garcia-Molina, Hector and Shivakumar, Shiva (2000). Crawler-Friendly Web Servers. In proceedings of Workshop on Performance and Architecture of Web Servers.

[8] Cho, J. and Garcia-Molina, H. (2000). Synchronizing a database to improve freshness. In Proceedings of ACM International Conference on Management of Data (SIGMOD), pages 117-128, Dallas, Texas, USA.

[9] Cho, J. and Garcia-Molina, H. (2003). Effective page refresh policies for web crawlers. ACM Transactions on Database Systems, 28(4).

[10] Cormen, Leiserson, Rivest, and Stein (2001).Introduction to Algorithms, MIT, ISBN 0-262-03293-7.

[11] Ezz Hattab, (2007). Updating Search Engines Using Meta-Updates. Int. Arab J. Inf. Technol. 4(4): 301-306.

[12] Gilmore, W. Jason (2006). Beginning PHP and MySQL 5: From Novice to Professional, Apress, ISBN 1590595521.

[13] Inktomi, (2000). Web surpassed one billion documents. URL:http://www.inktomi.com/new/press/billion.html.

[14] Jakob Nielsen (2005). "Reviving Advanced Hypertext". Retrieved on 2007-06-16.

[15] Levene, Mark (2005). An Introduction to Search Engines and Web Navigation. Pearson. ISBN10: 0321306775.

[16] Marc Najork and Janet L. Wiener, (2001). Breadth-first crawling yields high-quality pages. In Proceedings of the Tenth Conference on World Wide Web, pages 114–118, Hong Kong. Elsevier Science.

[17] Netcrawler project http://www.codeproject.com/KB/IP/Crawler.aspx.

[18] Qingzhao Tan, Ziming Zhuang, Prasenjit Mitra, C. Lee Giles, (2007). A Clustering-Based Sampling Approach for Refreshing Search Engine's Database.

[19] S. Lawrence and C. Giles. (1999). “Accessibility and distribution of information on the Web”. Nature, Vol. 400: pp.107–109.

Appendices

Appendix A – Tools code

A.1 The Proposed Approach Sample Calculation Algorithm

```
void calcFrequencySample(string crawlingDate)
        {
            try
            {
                OleDbConnection MyConnection1 = new
OleDbConnection(@"provider= Microsoft.Jet.OLEDb.4.0; Data Source =
crawler_records.mdb");
                double totalChangeRates = 0;
                int counter = 0;
                int samplingPages = int.Parse(samplingPagesBox.Text);

                string sql = "SELECT change_rate,site_domain FROM
frequency_greedy_change_rate WHERE crawling_date LIKE '" + crawlingDate
+ "'";
                OleDbCommand cmd = new OleDbCommand(sql);
                cmd.Connection = MyConnection1;
                MyConnection1.Open();
                OleDbDataReader thisReader = cmd.ExecuteReader();
                while (thisReader.Read())
                {
                    totalChangeRates +=
double.Parse(thisReader["change_rate"].ToString());
                }
                thisReader.Close();
                MyConnection1.Close();

                //sql again
                MyConnection1.Open();
                thisReader = cmd.ExecuteReader();
                while (thisReader.Read())
                {
                    domainSamplesArr[counter, 0] =
thisReader["site_domain"].ToString();
                    int sampleSize =
int.Parse((Math.Floor((double.Parse(thisReader["change_rate"].ToString(
)) / totalChangeRates) * samplingPages)).ToString());
                    if (sampleSize <5)
                        sampleSize = samplingPages / 20;
                    domainSamplesArr[counter++, 1] =
sampleSize.ToString();
                }
                thisReader.Close();
                MyConnection1.Close();
            }
            catch (Exception e)
            {
                log_errors(e.ToString());
            }
        }
```

A.2 The Greedy Sample Calculation Algorithm

```
void calcGreedySample(string crawlingDate)
        {
            try
            {
                OleDbConnection MyConnection1 = new
OleDbConnection(@"provider= Microsoft.Jet.OLEDb.4.0; Data Source =
crawler_records.mdb");
                double totalChangeRates = 0;
                int counter = 0;
                int samplingPages = int.Parse(samplingPagesBox.Text);

                string sql = "SELECT change_rate,site_domain FROM
greedy_change_rate WHERE crawling_date='" + crawlingDate + "'";
                OleDbCommand cmd = new OleDbCommand(sql);
                cmd.Connection = MyConnection1;
                MyConnection1.Open();
                OleDbDataReader thisReader = cmd.ExecuteReader();
                while (thisReader.Read())
                {
                    totalChangeRates +=
double.Parse(thisReader["change_rate"].ToString());
                }
                thisReader.Close();
                MyConnection1.Close();

                //sql again
                MyConnection1.Open();
                thisReader = cmd.ExecuteReader();
                while (thisReader.Read())
                {
                    domainSamplesArr[counter, 0] =
thisReader["site_domain"].ToString();
                    int sampleSize
=int.Parse((Math.Floor((double.Parse(thisReader["change_rate"].ToString
()) / totalChangeRates) * samplingPages)).ToString());
                    if (sampleSize <5)
                        sampleSize = samplingPages/20;
                    //else if (sampleSize > samplingPages/5)
                        //sampleSize = samplingPages/10;
                    domainSamplesArr[counter++, 1] =
sampleSize.ToString();
                }
                thisReader.Close();
                MyConnection1.Close();
            }
            catch (Exception e)
            {
                log_errors(e.ToString());
            }
        }
```

Appendix B - Curriculum Vitae

Sakher Khaleel Al-Qaaideh
Phone : +962 / 79 /
7153300 Email :
sakhr83@hotmail.com

EDUCATION

Middle East University for Graduate Studies , Amman – Jordan Master Degree Student of Computer Information Systems (final semester)	2008
Mutah University , Karak – Jordan Bachelor's of Computer Science	Jan 2005

Experiences

EtQ, Amman – Jordan Worked as web developer including tasks to write SQL queries, worked in the custom department where I implemented custom modifications to the product (Reliance), and fixed bugs in the product after being used by customers.	Dec 2007 – Sep 2008
Maktoob.com, Amman – Jordan I worked as web developer, I developed web sites using PHP (www.quran.maktoob.com), maintained VBulletin forums (www.sh3bwah.maktoob.com)	Sep 2007 - Nov
Jordan International Police Training Center, Amman – Jordan I maintained MS Access database including designing forms, reports and writing SQL queries. Then I moved to another department where i developed a PHP web site that contains curriculums to be studied by the trainees, stored as text in MS Access database and materials in MS word and PowerPoint files, the project name was IMP, web site was accessible internally in the local network, copy of the web is available.	Jun 2006 - Aug 2007
National Information Technology Center, Amman – Jordan I worked in the knowledge stations project (www.ks.jo), I was a computer skills trainer in Jabal Bani Hamida knowledge station in Madaba, I was preparing trainees for the ICDL certificate exam.	Oct 2005 – May 2006

SKILLS

Web development and design
PHP

ASP
JavaScript
xHTML
AJAX
Web servers: IIS, Apache and Tomcat Databases:
MySQL, SQL Server and MS Access Database
tools: PhpMyAdmin, EMS Manager and
MySQL GUI Tools Development
tools: Dreamweaver Forums:
vBulletin
Content Management Systems: Drupal
Search Engine Optimization (SEO) skills
- Meta tags
- Pages links
- Google analytics
- Sitemaps

Software development:
C# .Net
Java
Operating systems:
Windows
Linux - Fedora
Networking: design and setup LANs
Flash: swish 2
Microsoft office skills
Communication skills

LANGUAGES

English – Excellent Arabic –
Mother language

COURSES, CERTIFICATES

ICDL, certified
A+, Hashemite university course

PORTFOLIO

www.asesms.com
Amman Stock Exchange SMS web site is an e-commerce implementation, it provides services for getting SMS alerts on mobile phones for stock prices, communicating with bulk SMS gateway using API, and another API for accepting online payments, used PHP and MySQL database to develop this web site.

www.banihamida.net

A social network web site, users of the web site can create accounts and fill their information in profiles, home page shows topics written by members and they can add their comments to it, provides some services like prayer times and weather conditions and poll module. Users also can get web site notifications via email; simple messaging system is added to let members send private messages, used PHP and MySQL database to develop this web site, used my SEO skills to bring large amount of traffic from search engines to the web site.

IMP, Web version, MS Access version (Available upon request)
This web site was used in internal network to let instructors get materials for teaching lessons, lessons contained some description as text in MS Access database and materials on MS Word and MS PowerPoint files, accounts were used to authorize access to the classified web site and to download files, AJAX technology used extremely to add many features, lessons was stored in MS Access database connected to PHP pages using ODBC.