Updating Search Engines Using Meta-Updates

Ezz Hattab
Faculty of IS and Technology, Arab Academy for Banking and Financial Sciences, Jordan

Abstract: *Web search engines provide an extremely valuable service by indexing web content. However, much of this content is fluid; it changes, moves, and occasionally disappears, which leads to novel challenge to keep search engines up-to-date. This paper investigates how to keep search engines up-to-date by proposing meta-updates technique. Meta updates keep useful information about the behavior of a page to be provided to the spider of the search engine. The proposed technique saves many overheads in complex probabilities calculation suggested in the related works.*

Keywords: *Webcontent management, web content up dates, search engines freshnes.*

Received January 25, 2006; accepted September 14, 2006

1. Introduction

It would be inappropriate to buy seven copies of a daily newspaper in advance to save shopping time for one week! The improper part is that most of the newspaper content has a lifetime that is one day, thus the content in the next day will be useless. Correspondingly, web as any other media holds information with a lifetime in which it is useful. For example, events (e. g., call for papers, items discounts…etc) with a restricted deadline like *d* will be stale information after *d*.

Regrettably, due to occasional indexing, current search engines are unable to maintain the updated web information. Web crawling is a challenging task since it involves interacting with hundreds of thousands of web servers, which needs massive processing and data transferring that may take days or even weeks [1]. Consequently, it takes up to six months for a newborn page to be indexed by popular web search engines and the data that are indexed by the search engines are often stale [8].

This fact leads to a mismatch between the actual published pages at web sites and the corresponding indexed pages at the search engines. For example, let a page *P* indexed by the search engine *S* with content ς_1 at time τ_1, and let ε a short period of time that is much less than the time needed to have an opportunity for another indexing. Suppose that at $\tau_1 + \varepsilon$, the content of the page *P* has been updated to content ς_2 as illustrated in Figure 1. During that period, the page *P* will be out-of-date for a time interval of length $r_i = \tau_2 - (\tau_1 + \varepsilon)$ and thus any user query about ς_2 issued by that search engine (after ε) will not be able to see the new content of page *P*.

The user, as a result, *may* get irrelevant information to his query. Thus, a new issue of relevancy is the validity of the information passed by the search engine to the user queries.

This paper is organized as follows. Section 2 discusses the related work. Section 3 investigates how to keep search engines up-to-date by examining a kind of metadata. Section 4 introduces the architecture of the search engine crawler that parses the proposed meta-updates tag of change-frequency. Finally, section 5 concludes the work.

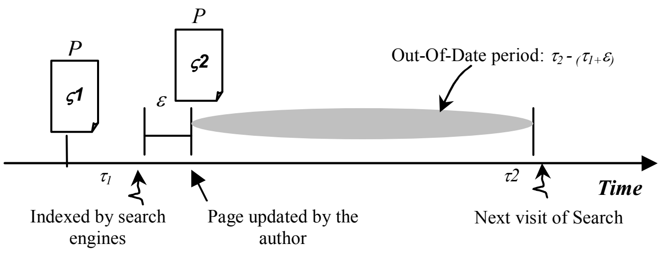

P
ς1
P
ς2
ε
Out-Of-Date period: τ2 - (τ1+ε)
τ1
τ2
Time
Indexed by search engines
Page updated by the author
Next visit of Search

Figure 1. Mismatches period between search engines and web servers.

2. Related Works

Most of previous work [2, 4, 6] assumed that search engines are pulling updates from web servers. This is by estimating the frequency of changes and using it as a heuristic for a re-indexing process in the future.

Coffman *et al.* [6] studied how often a crawler should visit a page when it knows how often the page changes. They built a theoretical model to minimize the fraction of time pages spend out-of-date. Also assuming Poisson page change processes and a general distribution for page access time, they similarly show that optimal results can be obtained by crawling pages as uniformly as possible.

They discussed the crawling process from a theoretical viewpoint, comparing it to the polling systems of queuing theory, that is, and multiple queue-single server systems. Specifically, they assumed independent Poisson page change processes, and a general distribution for the page access time τ (i. e., $\tau_2 - \tau_1$). They showed that, if τ is decreased in the increasing convex ordering sense, then the out-of-date (r_i) is decreased for all i under any scheduling policy, and that, in order to minimize expected total obsolescence time of any page, the accesses to that page should be as evenly spaced in time as possible. On the other hand, [2] assumed a local collection $S = \{e_1, ..., e_N\}$ of N pages. Then they defined the *freshness* F of the collection as follows:

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

where e_i is the local page, t is the time and *up-to-date* means that the content of a local page equals that of its real-world counterpart. Thus, the freshness of the local collection S at time t is:

$$F(S;t) = \frac{1}{N}\sum_{i=1}^{N} F(e_i;t)$$

The *freshness* is the fraction of the local collection that is up-to-date. For instance, $F(S; t)$ will be one if all local pages are up-to-date, and $F(S; t)$ will be zero if all local pages are out-of-date.
Further, they defined the *age* concept in order to capture how old a local page e_i at time t is, which is expressed as follows:

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

Then, the age of the local collocation S is

$$A(S;t) = \frac{1}{N}\sum_{i=1}^{N} A(e_i;t)$$

The age of S recall the average "*age*" of the local collection. For instance, if all real-world pages changed one day ago and we have not refreshed them since, then $A(S; t)$ is one day.

The above definitions are given for time t to approximate the average of freshness over a long period of time $t \to \infty$ then F is redefined:

$$\overline{F}(e_i) = \lim_{t\to\infty} \frac{1}{t}\int_0^t F(e_i;t)dt$$

$$A(S) = \lim_{t\to\infty} \frac{1}{t}\int_0^t A(S;t)dt$$

As we have noted, most studies make the assumption that pages change at a variable rate, which may be approximated by a Poisson distribution (in the second stage of their study, Cho and Garcia-Molina [5] assume the broader gamma distribution for this change rate). Moreover, they prove that their particular model is valid for any distribution, and conclude that when pages change at varying rates, it is always better to crawl these pages at a uniform rate, that is ignoring the rate of change, than at a rate, which is proportional to the rate of change. To maximize freshness they found a closed form solution to their model, which provides an optimal crawling rate, which is better than the uniform rate.

These results were all derived for a batch or periodic crawler, where a fixed number of pages is crawled in a given time period. These pages are used to update a fixed size repository either by replacing existing repository pages with newer versions or by replacing less important pages with those deemed to be of greater importance.

In [5], researchers devised the architecture for an incremental crawler, and examined the use of an incremental versus a batch crawler under various conditions. Crawling a subset (720,000 pages from 270 sites) of the web daily, they determined statistics on the rate of change of pages from different domains. They found, for example, that for the sites in their survey, 40% of pages in the *.com* domain change daily in contrast to *.edu* and *.gov* domains where more than 50% of pages did not change in the four months of the study. They showed that the rates of change of pages they crawled can be approximated by a Poisson distribution that change more often than daily or less often than four monthly are inaccurate.

In [2, 10], researchers removed the memoryless assumption implicit in a poisson process by modeling the web changes as a renewal process. They further defined a *freshness* metric to characterize how up-to-date a local information repository is computed to the web. This is some what undermined based on an extensive subset of the web by Brewington *et al.* showing that most web pages are modified during US working hours, that is, 5 am to 5 pm, Monday to Friday.

In [10], researchers estimated the cumulative probability function for the page age in days on a log scale as shown in Figure 2 and they estimate the cumulative probability of mean lifetime in days shown in Figure 3.

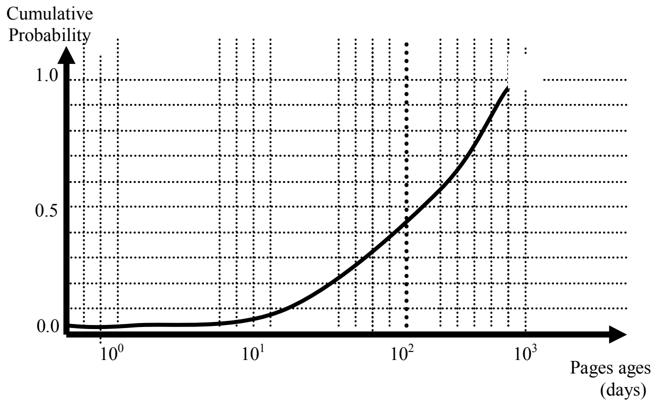


Figure 2. Cumulative probability distribution of page age in days.

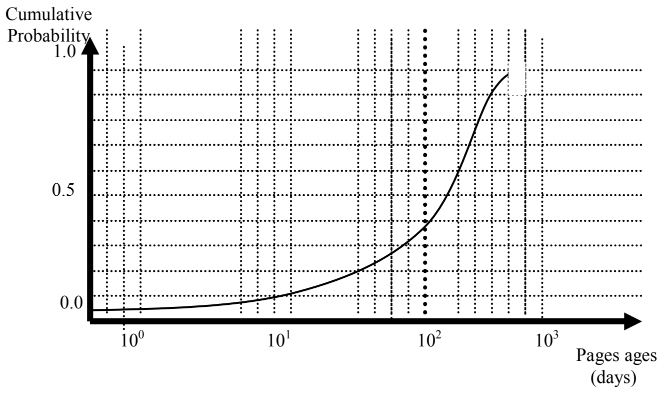


Figure 3. Cumulative probability distribution of mean lifetime e in days.

3. The Proposed Approach: Meta Updates

In [10], researchers showed that to be sure with 95% probability that a randomly chosen page is fresh up to a day ago, then web of 800M pages need a re-indexing period of 8.5 days, and a re-indexing period of 18 days is needed to be sure that the repository copy of a random page is fresh up to one week ago with 95% probability.

At the same time, Inktomi and NEC Research Institute have completed a study that verifies that the web has grown to more than one billion unique pages [7].

According to the observations in [3, 7] and assuming a web page has an average size of 12 kilobytes, then a search engine should re-index the web every:

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

Obviously, the above simple calculation shows that such a scenario is impractical bearing in mind that the web size is doubling yearly [8].

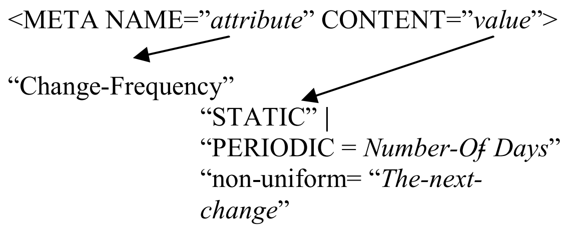
We propose a non-uniform re-indexing through coordination between web servers and search engines. By the "coordination" we mean that a web server informs a search engine about the next expected change. This will facilitate the next visit of the search engine crawler to the web server. We propose a kind of meta data to represent such coordination.

World Wide Web Consortium [9] proposed meta tags that describe page properties like title, author, keywords, descriptions, etc. Typically, crawlers of generic search engines (e. g., Google, Yahoo, Alta vista, etc) follow these meta tags before downloading web pages.

Based on meta tags, we have the chance to inform search engine crawlers about the next expected change of a page. Thus, we keep the index of search engines fresh with a minimum number of re-crawling. Further, we can avoid needless crawling of pages that change very slowly. Change frequency tag has three possible values:

1. Static value indicating that a page has very little probability to be changed. By default we recommend Static = 90 (i. e., 3 months), which is the expected re-crawling time of the typical search engines.
2. Periodic value indicating a page changes in a uniform rate.
3. A variable indicating a page changes in a non-uniform rate. Static and periodic values are provided by webmasters, while non-uniform values are estimated by software applications.

The proposed meta tag has the following general form:


<META NAME="*attribute*" CONTENT="*value*">
"Change-Frequency"
"STATIC" |
"PERIODIC = *Number-Of Days*"
"non-uniform= "*The-next-change*"

Example 1 shows various forms of the proposed meta tag.

Example 1:

Forms of "Change-Frequency" Meta tag
Changing frequency in uniform rates:

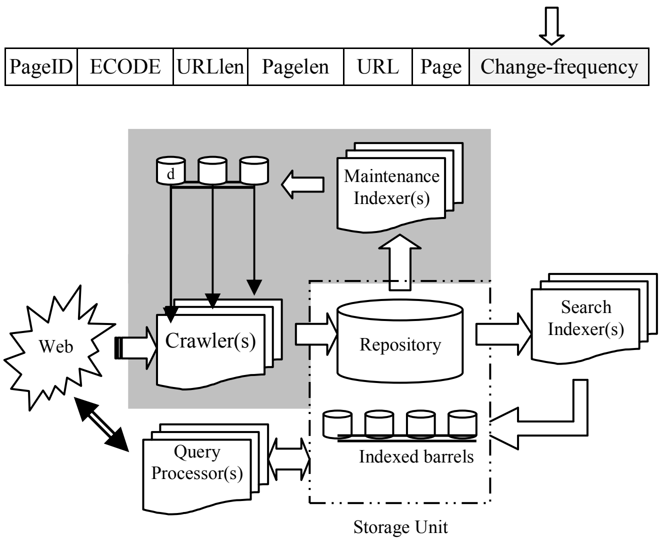
1. <META NAME="Change-Frequency" CONTENT="STATIC"> indicates that the current page is static and no need for re-indexing.
2. <META NAME="Change-Frequency" CONTENT="PERIODIC=1"> indicates that the current page is changing every day (e. g., a Newspaper).
3. <META NAME="Change-Frequency" CONTENT="PERIODIC=7"> indicates that the current page is changing every week (e. g., a weekly magazine)

Non-uniform frequency, thus only the next visit is provided:

<META NAME="Change-Frequency" CONTENT="non-uniform=<% Change_Frequency() %>">. Since the page changes in non-uniform frequency, the non-uniform value is generated dynamically by an application (e. g., ASP application server of Microsoft).

Accordingly, a search engine could index the pages for maintenance purposes using the available meta-update information. Roughly, pages could be indexed into barrels that hold pages' change daily, weekly, monthly…etc as illustrated in Figure 4-a. As shown, there are three barrels that hold pages change daily, weekly and monthly. Each barrel is assigned for one crawler(s) that performs the re-crawling process. The re-crawling process is described in an algorithmic form in Figure 4-b.

In this case, the database of the search engine is modified by adding a field of "change-frequency" that holds the value of the meta tag as follows:



(a)

```
Scheduled-URLs ← To-Do-List (Indexed-URLs)
Re-crawl (change-frequency)
{// according to Meta-updates found at d atabase of search engine
   While ( Ordered-URLs is not Empty and not Visited and Permissible)
     {
        Parse (Meta tags of HTML Head)
        Modify search-engine-database
        If-modified-since (last-crawl-date)
          {
            Download (URLs) {
               Get-Content (Page)
                Parse (Page)
                Extracted-URLs ← Extract-URLs-from-page (Page)
                Visited ← Add-to-visited-URLs (URLs)
                For each URL in Extracted-list {
                   Add to Scheduled-URLs
                }
            }
          }
}
```

(b)

Figure 4. (a) Search engine with periodical indexing barrels (b) The modified algorithm.

4. Experimental Evaluation

In this section, we build a search engine crawler that parses the proposed meta update tag of change-frequency and achieves the crawling process according to the algorithm shown in Figure 4. The results are compared with the ordinary crawler in order to evaluate crawling with meta update.

Details of the Experiment:

- *Local Web*: We have created local web that mirrors ten websites. This is done in order to be able to add the meta updates to web pages as we do not have such permission at the actual sites. This is achieved by launching a mirroring program that retrieves the specified URL. It can also retrieve recursively any link that a page references.
- *Instantiating Meta Updates*: To instantiate meta updates with static, periodic or proportional, we follow the following algorithm:

  1. We count the number of changes per unit of time *t* (i. e., 7 days). Log the value into the database (*count$_i$* = *number-of-changes*). The changes are detected daily by checking the last-modified date (i. e., HTTP header field).
  2. If *count* = 0, then we prefix this value by "Static" keyword and post it into the meta update tag.
  3. Otherwise, we post the value of "Periodic = *count*".
  4. We check the value of *count* in a period of time τ (where $\tau > t$). For example, in four weeks we have ($count_1$, $count_2$, $count_3$, $count_4$). If the values of counts vary (i. e., $count_1 \neq count_2 \neq count_3 \neq count_4$), we consider that the page changes in a non-uniform rate and post the last value to the meta update tag associated with "non-uniform = $count_i$" keyword.

- *Typical Crawling*: We launched the typical crawler (that is based on the algorithm presented previously in Figure 2). The crawler downloads pages into the local repository named "Typical_Repository". We set the typical crawler to fetch the page periodically every week regardless whether the page has been changed or not. This is to simulate the crawlers of typical search engines.
- *Crawling with Meta Update*: At the same time, we launched the crawler that considers the meta update tag of change-frequency. The crawler downloads the pages into a local repository named "Updated_Reposi-toy". The crawler is based on the algorithm shown in Figure 4. Obviously, the pages are downloaded according to the value of the change-frequency.
- *Freshness and Needless Crawling*: We log the last modified date for each page daily. This is to check whether the crawling process is well-suited for the actual change frequency or not. Thus, we examine the freshness and needless crawling.
- *Period of Experiment*: The monitoring period took six months.
- *Results*: Table 1 and Table 2 present an extracts of the statistics for both typical and meta updates respectively. Since the chosen web sites are very dynamic, we extracted statistics of a page that changes in a non-uniform rate (i. e., http://www.sun.com/index.html).

As noticed, there are various patterns that occurred in both cases. For example, in the typical crawling and on Day5/Week1, a change on the site occurred that switched the freshness of the site to 0. On Day1/Week3, a needless crawling occurred as the crawler visited the site with no need (the freshness = 1). While crawling with meta updates support, gives a relatively higher freshness rate, there occurs some mismatch in the dates (almost in hours) that caused both zero freshness on Day5/Week2 and needless crawling on Day3/Week2.

Table 3 and Table 4 illustrate the overall freshness for each web site in both typical and meta updates crawling, respectively.

Figure 5 shows a visual comparison between the typical crawling and the crawling that supports the meta-updates. It is obviously clear that the freshness is relatively better in the case of the meta-updates support.

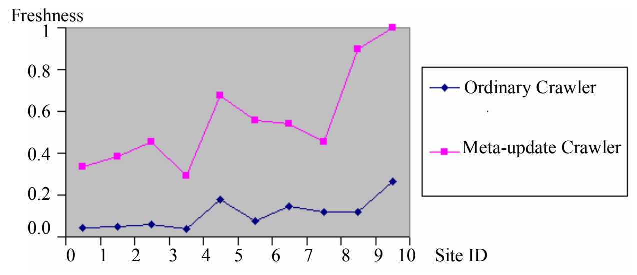


Figure 5. Typical (Ordinary) vs. meta-updates crawlers (freshness average).

There is a trade-off between freshness and costs of search engines and web servers. As depicted in Figure 6, there are many more crawling processes in crawling that supports meta-updates.

It is in the interest of web servers to support an efficient crawler refresh strategy because it can achieve higher freshness and could utilize the resources of search engines that are otherwise wasted in needless crawling to index newborn pages and thus leading to increase the indexed web size (i. e., 6%-12%) [8].

Table 1. An extract of statistics that are collected by typical crawling (1 means the page is fresh; 0 otherwise).

http://www.sun.com/index.html					
Week Number	**Day Number**	**Did the Page Change?**	**Is the Page Crawled?**	**The Freshness is:**	**Is it a Needless Crawling?**
Week1	Day1	Yes	Yes	1	No
	Day2	No	No	1	-
	Day3	No	No	1	-
	Day4	No	No	1	-
	Day5	Yes	No	0	-
	Day6	No	No	0	-
	Day7	No	No	0	-
Week2	Day1	No	Yes	1	No
	Day2	No	No	1	-
	Day3	No	No	1	-
	Day4	No	No	1	-
	Day5	No	No	1	-
	Day6	No	No	1	-
	Day7	No	No	1	-
Week3	Day1	No	Yes	1	Yes
	Day2	No	No	1	-
	Day3	Yes	No	0	-
	Day4	No	No	0	-
	Day5	Yes	No	0	-
	Day6	No	No	0	-
	Day7	No	No	0	-
…	…	…			

Table 2. An extract of statistics that are collected by meta updates crawling (1 means the page is fresh; 0 otherwise).

http://www.sun.com/index.html					
Week Number	**Day Number**	**Did the Page Change?**	**Is the Page Crawled?**	**The Freshness is:**	**Is it a Needless Crawling?**
Week1	Day1	Yes	Yes	1	No
	Day2	No	No	1	-
	Day3	No	No	1	-
	Day4	No	No	1	-
	Day5	Yes	Yes	1	No
	Day6	No	No	1	-
	Day7	No	No	1	-
Week2	Day1	No	No	1	No
	Day2	No	No	1	-
	Day3	No	Yes	1	Yes
	Day4	Yes	No	0	-
	Day5	No	No	0	-
	Day6	No	No	0	-
	Day7	No	Yes	1	No
Week3	Day1	No	Yes	1	Yes
	Day2	No	No	1	-
	Day3	Yes	No	1	-
	Day4	No	No	1	-
	Day5	Yes	No	1	-
	Day6	No	Yes	1	-
	Day7	No	No	1	-
…	…	…			

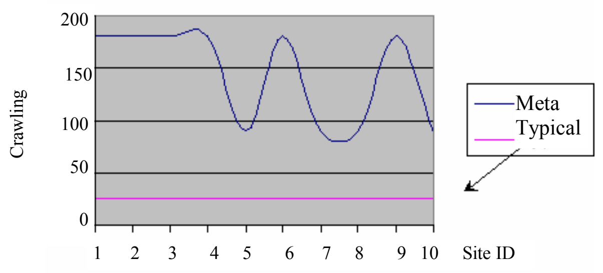


Figure 6. Typical vs. meta-updates crawlers (number of crawling).

Table 3. The results of crawling web pages using ordinary crawler (number of crawls are 24, i. e., one crawl per week).

The Site ID	**No. of Changes**	**Freshness**	**No. of Needless Crawling**	**Websites (Pages)**
1	540	0,045	8	www.oracle.com
2	470	0,051	4	www.ntua.gr
3	397	0,061	2	www.stanford.edu/
4	611	0,039	3	www.microsoft.com/
5	133	0,180	4	www.sun.com/
6	324	0,074	6	searchenginewatch.com
7	166	0,144	6	www.siliconvalley.com
8	198	0,121	4	usa.net
9	200	0,120	3	www.amazon.com/
10	90	0,266	10	www.harvard.edu/

5. Conclusion

In this paper, we proposed meta-updates in order to schedule the next visit of crawlers. We evaluated the proposed approach by an experiment that compares the freshness degree in both typical crawling, which is used in generic search engines, and the crawling with meta-updates support. The experiment showed that meta-updates improve freshness by an average of 45% in most cases. On the other hand, meta-updates crawling needs a lot of resources that entails a trade-off selection between freshness and search engines' resources.

Table 4. The results of crawling web pages using metadata crawler.

The Site ID	No. of Changes	Freshness	No. of Crawling	No. of Needless Crawling	Websites (pages)
1	540	0,334	180	2	www.oracle.com
2	470	0,383	180	1	www.ntua.gr
3	397	0,453	180	3	www.stanford.edu
4	611	0,294	180	2	www.microsoft.com
5	133	0,676	90	4	www.sun.com
6	324	0,555	180	3	searchenginewatch.com
7	166	0,542	90	2	www.siliconvalley.com
8	198	0,454	90	4	usa.net
9	200	0,900	180	2	www.amazon.com
10	90	1,000	90	1	www.harvard.edu

References

[1] Brin S. and Page L., "The Anatomy of a Large-Scale Hypertextual Web Search Engine," *Computer Networks and ISDN Systems*, vol. 30, no. 1-7, pp. 107-117, 1998.

[2] Brewington B. E. and Cybenko G., "How Dynamic is the Web?" *Computer Networks*, vol. 33, no. 1-6, pp. 257-276, 2000.

[3] Brewington B. and Cybenko G., "Keeping up with the Changing Web," *IEEE Computer*, vol. 333, no. 5, pp. 52-58, 2000.

[4] Cho J. and Garcia-Molina H., "Synchronizing a Database to Improve Freshness," *in Proceedings of the SIGMOD*, USA, pp. 117-128, 2000.

[5] Cho J. and Garcia-Molina H., "The Evolution of the Web and Implications for an Incremental Crawler," *in Proceedings of the 26th International Conference on Very Large Databases*, Stanford, CA, pp. 200-209, 2000.

[6] Coffman E. G., Liu Z., and Weber R. R., "Optimal Robot Scheduling for Web search Engines," *Technical Report RR-3317*, Institut National de Recherche en Informatique et en Automatique (INRIA), France, pp. 19, 1997.

[7] Inktomi, "Web Surpassed One Billion Documents," available at: http://www.inktomi.com/new/press/billion.html, 2000.

[8] Lawrence S. and Giles C., "Accessibility and Distribution of Information on the Web," *Nature*, vol. 400, pp.107-109, 1999.

[9] W3C Consortium: Meta Elements, available at: URL: http://www.w3.org/TR/REC-html32, 1997.

[10] Wills C. and Mikhailov M., "Towards a Better Understanding of Web Resources and Server Responses For Improved Caching," *Computer Networks*, vol. 31, no. 11-16, pp. 231-1243, 1999.

Ezz Hattab has been awarded a scholarship from the EU to pursue his PhD in information technology. His PhD research was part of the Europe research project "The Webminer", in which he proposed numerous algorithms and techniques that handle web information retrieval and search. Currently, he is working at Arab Academy for Banking and Financial Sciences. Previously, he held various academic positions at Applied Science University, Amman Ahlyyia University, and Amman Arab University. He is an active member of numerous professional and scientific societies, including the Arab Society of Computers (ASC), and Jordanian Society of Computers (JSC). He has 23 publications and 4 books in the area of Information Technology, e-business, and web applications. He is a member of the technical committee of The International Arab Journal of Information Technology (IAJIT), EBEL, e-Jordan and an associate editor of the International Journal of Mobile Learning and Organization (IJMLO). He is a member of the high supervision committee of Management Information Stream at the Ministry of Education.