Simulation Study of Language Specific Web Crawling

Kulwadee Somboonviwat[†] Takayuki Tamura[†,††] Masaru Kitsuregawa[†]

[†] *Institute of Industrial Science, University of Tokyo*

[††] *Information Technology R&D Center, Mitsubishi Electric Corporation*

E-mail: [†] *{kulwadee,tamura,kitsure@tkl.iis.u-tokyo.ac.jp}*

Abstract

The Web has been recognized as an important part of our cultural heritage. Many nations started archiving national web spaces for future generations. A key technology for data acquisition employed by these archiving projects is web crawling. Crawling cultural and/or linguistic specific resources from the borderless Web raises many challenging issues. In this paper, we propose the language specific web crawling and evaluate the language specific crawling strategies on the web crawling simulator.

1. Introduction

The increasing importance of the World Wide Web as being a part of our cultural heritage is a key motivation for many nations to start archiving the Web for future generations. One of the key technologies for automatic accumulation of web pages is *web crawling*.

The web crawling approach currently used in crawling national web space is the domain-name based restriction approach, e.g. crawling pages under the '.th' top-level domain for Thai web space. Though this approach can be implemented easily, its major drawback is the failure to obtain relevant pages residing in other domains such as '.com', '.net'. We propose a language based restriction approach for the national web space crawling, which will be called a *language specific web crawling*.

In this paper, we propose and evaluate two crawling strategies for crawling language specific resource i.e. a referrer's relevance based strategy and a distance based strategy. Both strategies are designed based on the assumption that there is a *language locality* on the Web.

To avoid social and technical difficulties associated with the evaluation of web crawling strategies, we will evaluate crawling strategies on a web crawling simulator. We use data from the actual crawl logs as an input to the web crawling simulator. Logically, the web crawling simulator constructs a virtual web space from the information available in the crawl logs. During a simulation process, the simulator generates requests for web pages to the virtual web space, according to the user-specified web crawling strategy.

We evaluated the referrer's relevance based strategy and the distance based strategy on the Thai dataset (14 million URLs). From the simulation results, both strategies achieve better performance than the breadth-first strategy which does not utilize the language locality in the selection of crawl paths. We also propose a variation of the distance based strategy i.e. the limited distance strategy which allows the user to adjust the number of URL queues according to available system resources by varying a distance parameter *N*.

The paper is organized as follows: Section 2 presents related work. Section 3 describes the essences of language specific web crawling. In Section 4, the architecture of the web crawling simulator is explained. Subsequently, in Section 5, we report the simulation results and evaluations. Finally, Section 6 concludes the paper.

2. Related work

A Web crawler is a tool for automatically gathering web pages. A general-purpose web crawler works by iteratively downloading web pages, and following hyperlinks to create a local repository of web pages. The explosion of information available on the Web poses scalability problems – both in terms of storage and performance – on the general-purpose web crawlers. Some interesting methods proposed are those of *Fish Search* [4] and *focused crawling* [1]. These methods try to selectively download web pages that are relevant to a specific topic. Thereby, the amount of network traffic and downloads can be reduced.

2.1. Focused crawling

Focused crawling was proposed by Chakrabarti et al. [1]. The goal of a focused crawler is to selectively seek

out pages that are relevant to the predefined topics of interest. The idea of focused crawling is based on an assumption that there is a *topical locality* in the Web i.e. web pages on the same topic are likely to be located near on the Web graph.

The focused crawling system has three main components: a *classifier* which determines relevance of crawled pages to decide on link expansion, a *distiller* which identifies *hubs*, i.e. pages with large lists of links to relevant web pages, and a *crawler* which is controlled by the classifier and the distiller.

We will adapt the idea of focused crawling to language specific web crawling. The following section discusses language specific web crawling in more detail.

3. Language specific web crawling

The first version of a language specific web crawler has two main components: a **crawler** and a **classifier**. The crawler is responsible for the basic functionality of a web crawling system e.g. downloading, link extraction, URL queue management, and imposing a crawling strategy. The classifier determines relevance of crawl pages.

The relevance of a given page can be determined from its language. If a web page is written in the target language then its relevance score will be equal to 1 (relevant), otherwise its relevance score will be equal to 0 (irrelevant). Our classifier determines the language of a given web page from its character encoding scheme. There are two methods for automatically detecting the character encoding scheme of a given web page i.e. 1) checking the HTML's META tag for the charset property, and 2) employing the charset detector tool such as the Mozilla Charset Detectors [5].

In this paper, we will test the language specific web crawling strategies on Thai dataset. Unfortunately, the Mozilla Charset Detectors does not provide support for Thai language. So, we will rely on the information provided in the charset property of the HTML's META tag for the determination of web pages' languages.

In language specific web crawling, we assume the existence of *language locality* on the Web, i.e. web pages written in the same language tends to be located near on the Web graph. From the language locality, it can be implied that

Implication 1: A web page is likely to cite another web page written in the same language.

Implication 2: A web page is likely to locate near other web pages written in the same language.

By exploiting these implications, we have developed two language specific crawling strategies: 1) a referrer's relevance based strategy, and 2) a distance based strategy.

The **referrer's relevance based strategy** is based on the implication 1 stated earlier. In this strategy, the crawler will follow links extracted from web pages which are written in the target language (i.e. relevant referrer) before links extracted from web pages written in other languages (i.e. irrelevant referrer).

The **distance based strategy** is based on the implication 2. In this strategy, the crawler will follow links extracted from web pages with a minimum distance from the latest relevant referrer page (represented by *n*). Figure 1 gives an example of how to determine the distance *n*. This strategy can be implemented by maintaining a separate queue for each set of URLs having the same value of *n*.

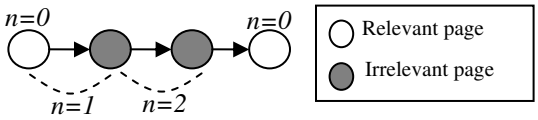

Figure 1. Distance to the latest relevant referrer page

4. Web crawling simulator

Evaluating crawling strategies under the real dynamic web environment poses the following issues.

- *Social issues:* It is an ill-manner to let the buggy crawler annoy the remote servers. We need to test a crawler extensively before launching it into the Web.

- *Technical issues:* Because the web is very dynamic, it is impossible to ensure that all strategies are compared under the same condition.

- *Resource issues:* Web crawling consumes time, network bandwidth and storage space.

To avoid these issues, we will evaluate our crawling strategies on the web crawling simulator. The web crawling simulator is a trace-driven simulation system which utilizes crawl logs to create a virtual web space for the crawler.

The architecture of the crawling simulator is as shown in Figure 2. The crawling simulator consists of the following components: a simulator, a visitor, a classifier, an observer, URL queue(s), crawl logs, and a link database.

A **simulator** initializes and controls system execution. A **visitor** simulates operations of a crawler i.e. managing the URL queue, downloading of web pages, and extracting new URLs. A **classifier** determines relevance of a given page by applying methods described in the previous section. An **observer** is an implementation of the web crawling strategy to be evaluated.

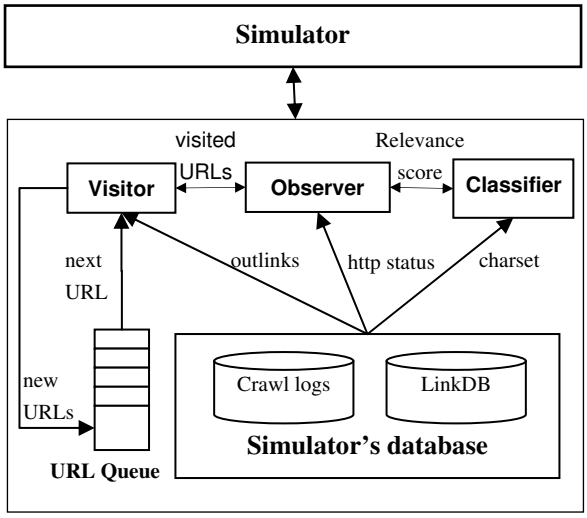

Figure 2. Architecture of the Web Crawling Simulator

5. Experiments

In this section, we describe the experiments conducted on the web crawling simulator, and report the evaluation results for our language specific crawling strategies.

5.1. Thai Dataset

Our dataset was obtained by actually downloading web pages from the Internet. We started our crawl from some numbers of well-known Thai web directories, and limited our crawl scope by imposing a limit on the depth of crawl paths. Table 1 summarizes the characteristics of the Thai dataset.

Table 1. Summary of dataset's characteristics

Content/Transaction Type	Number of URLs
Thai HTML pages	1,489,126
Non-Thai HTML pages	2,536,556
Non-HTML contents	360,404
Non-successful transaction	9,433,693
HTTP/HTML Redirection	420,946
Total	**14,240,725**

5.2. Results

Figure 3 and Figure 4 respectively show harvest rate and crawl coverage graphs of a referrer's relevance based strategy, a distance based strategy, and a breadth-first strategy.

From the harvest rate graph in Figure 3, the referrer's relevance based and the distance based strategies achieve better performance than the breadth-first strategy which does not exploit the language locality property in selecting the crawl paths.

According to the harvest rate graph, the referrer's relevance based strategy and the distance based strategy seem to have the same behavior. However, if we look at the coverage graph in Figure 4 more carefully, during the crawl progress between 2.5e+06 and 6e+06 pages, there is little difference i.e. the distance based strategy obtains more relevant pages than the referrer's relevance based strategy.

Figure 5 shows a graph plotting the number of crawled URLs classified by distance from the latest relevant referrer page (n). From this graph, it can be seen that most of the relevant crawled pages (about 92% out of 1,489,126 relevant pages) are pages with n =1 i.e. pages which are referred to by the relevant referrer. And, there are about 75K, 25K, 8K, and 4K relevant pages with distance n=2, n=3, n=4, and n>=5 respectively.

After URLs with n=1 (i.e. URLs extracted from relevant referrers) are exhausted, the distance based strategy will select next URLs to be downloaded from the queue for URLs with n=2. But the referrer's relevance based strategy will behave similar to a breadth-first strategy. This is the reason why the distance based strategy could obtain more relevant pages during the crawl progress between 2.5e+06 and 6e+06 pages which is approximately the point where URLs with n=1 are exhausted from the queue (from Figure 5, there are about 2.2e+06 URLs with n=1).

From the graph in Figure 5, almost all URLs with n>=5 are irrelevant. Thus, it may not be worth maintaining queues for these sets of URLs. The distance based strategy may be adapted to keep only URLs with n<=N, where N is a parameter specified by the user. We will call this modified version as a "limited distance based" strategy. Figure 6 shows the aggregate size of the URL queue (i.e. total number of enqueued URLs) for the limited distance based strategy with N=2 and N=3 in comparison with the distance based strategy and the referrer's relevance based strategy. From the graph, the size of the URL queue varies with the parameter N. Therefore, the URL queue size can be controlled by setting an appropriate value of the distance parameter N.

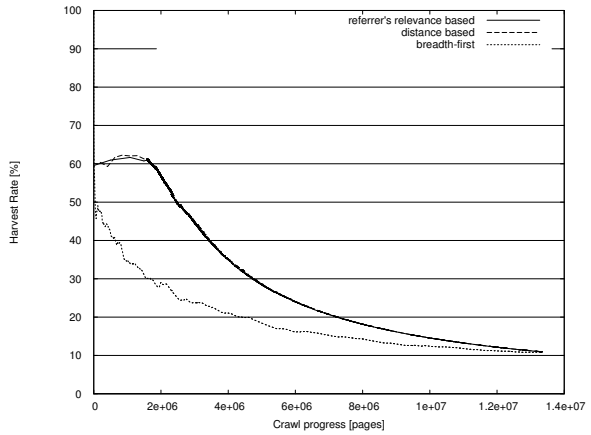

Figure 3. Harvest rate

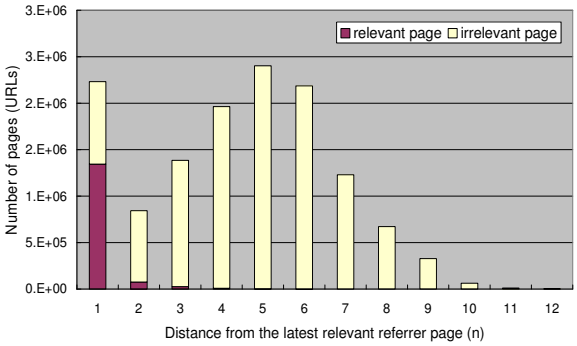

Figure 5. Number of relevant/irrelevant URLs classified by distance from the latest relevant referrer page (*n*)

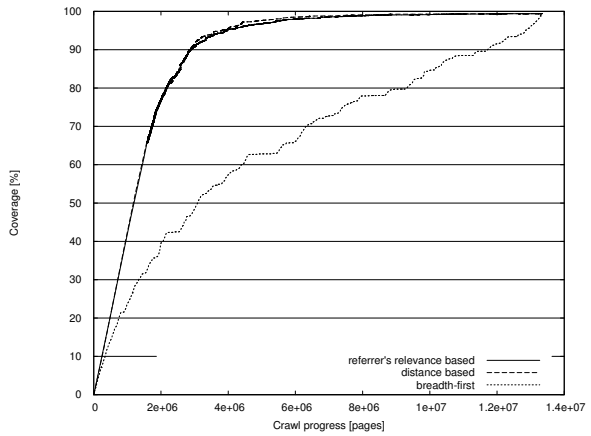

Figure 4. Coverage

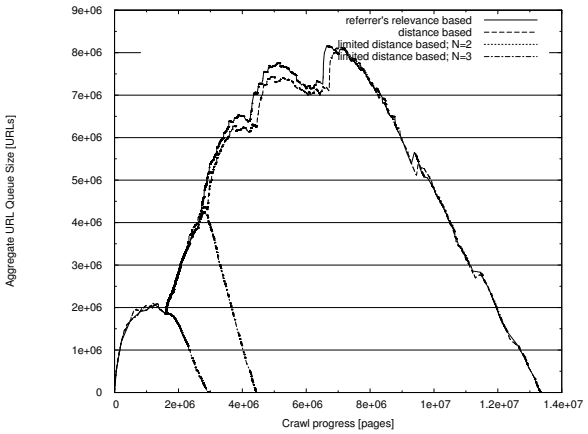

Figure 6. Aggregate URL queue size

6. Conclusion

In this paper, we have proposed a language specific web crawling for supporting the national and/or cultural web archiving projects. The idea of focused crawler (Chakrabarti et.al.[1]) was adapted to language specific web crawling with an assumption that there exists "language locality" in the Web.

We exploited the language locality property in the design of our crawling strategies: a referrer's relevance based and a distance based strategies. The evaluation of these strategies was done on the web crawling simulator to avoid technical and social problems occurred when conducting the evaluation on the real Web. We used the actual crawl logs corresponding to the crawl of a portion of Thai web space as our dataset.

From the experimental results, our proposed strategies which exploit the language locality in selecting the crawl paths achieve higher performance than a naïve breadth-first crawling strategy.

7. References

[1] Soumen Chakrabarti, Martin van den Berg, and Byron Dom, "Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery". Computer Networks, 31(11-16):1623-1640, 1999.

[2] Soumen Chakrabarti, Martin van den Berg, and Byron Dom, "Distributed hypertext resource discovery through examples". Proc. of 25th Int. Conf. on Very Large Data Bases, pages 375-386, September 1999.

[3] J. Cho, H. Garcia-Molina, and L. Page, "Efficient Crawling Through URL Ordering". Computer Networks, 30(1-7):161-172, 1998.

[4] P. De Bra, R. Post. "Searching for Arbitrary Information in the WWW: the Fish-Search for Mosaic", WWW Conference, 1994.

[5] The Mozilla Charset Detectors. Available at http://www.mozilla.org/projects/intl/chardet.html