

# On URL Normalization<sup>†</sup>

Sang Ho Lee<sup>1</sup>, Sung Jin Kim<sup>2</sup>, and Seok Hoo Hong<sup>1</sup>

<sup>1</sup> School of Computing, Soongsil University, Seoul, Korea  
shlee@comp.ssu.ac.kr, vonsaki@korea.com

<sup>2</sup> School of Computer Science and Engineering,  
Seoul National University, Seoul, Korea  
sjkim@oops.snu.ac.kr

**Abstract.** Since syntactically different URLs could represent the same resource in WWW, there are on-going efforts to define the URL normalization in the standard communities. This paper considers the three additional URL normalization steps beyond ones specified in the standard URL normalization. The idea behind our work is that in the URL normalization we want to minimize false negatives further while allowing false positives in a limited level. Two metrics are defined to analyze the effect of each step in the URL normalization. Over 170 million URLs that were collected in the real web pages, we did an experiment, and interesting statistical results are reported in this paper.

## 1 Introduction

A Uniform Resource Locator (URL) is a string that represents a web resource (here, a web page). A URL is composed of five components: scheme, authority, path, query and fragment [1]. Given a URL, we can identify a corresponding web page in the World Wide Web (WWW). If some URLs locate the same web page in the WWW, we call them equivalent URLs in this paper. Syntactically identical URLs are certainly equivalent, but there are cases in which syntactically different URLs represent the same web page (hence, are equivalent).

One of the most common operations on URLs is simple comparison to determine if two URLs are equivalent without using the URLs to access their web pages. For example, a web crawler (web robot) encounters millions of URLs during collecting web pages in the WWW, and it needs to determine if a newly found URL is equivalent with ones of the already-crawled URLs to avoid duplication of request actions. Syntactically different URLs that are indeed equivalent give rise to a large amount of processing overhead; a web crawler requests, downloads, and stores the same page repeatedly, consuming unnecessary network bandwidth, disk I/Os, disk space, and so on. Extensive normalization [2][5] prior to comparison of URLs is often used by many web crawlers to prune a search space or reduce duplication of request actions.

---

<sup>†</sup> This work was supported by Korea Research Foundation Grant. (KRF-2004-005-D00172)

The details of URL normalization methods that have been used (or are currently used) in real web crawlers have not been described in the literature yet. The URL normalization issues have not drawn technical interests successfully in the research communities yet, in spite of their importance in web application developments. Moreover, the existing URL normalization methods have been developed rather on the basis of developers' personal heuristics, not from an extensive analysis of experimental simulation. We need to approach this matter in an analytic and systematic way.

It is possible to determine that two URLs are equivalent, but it is never possible to be sure that two URLs represent different web pages. For example, an owner of two different domain names could decide to serve the same resource from both, resulting in two different URLs. Any URL normalization methods could cause occurrences of so-called false negative (determining equivalent URLs to be not equivalent, hence not being able to transform equivalent URLs into a syntactically identical string). People want to use URL normalization methods that transform equivalent URLs into an identical string as much as possible (hence reduce the occurrence rate of false negatives as low as possible).

There are on-going efforts to define the URL normalization in the standard body [1]. The standard URL normalization defines a number of steps to transform URLs into the canonical form of URLs, so that it helps determine whether given two URLs are equivalent. The standard URL normalization transforms URLs that are determined to be equivalent, into the syntactically identical canonical form. The standard URL normalization is designed to minimize false negatives while strictly avoiding false positives (determining non-equivalent URLs to be equivalent); it never transforms non-equivalent URLs into a syntactically identical string. However, we notice that in reality there are cases in which we can minimize false negatives significantly while allowing false positives in a very limited way.

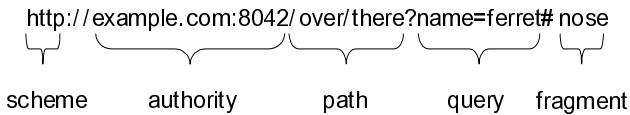
This paper considers the three URL normalization steps that are beyond the standard URL normalization. Discussed three steps are the case sensitivity at the path component of a URL, the last slash symbol in the path component of URLs, and the designation of a default page. The idea behind our approach is that in the URL process we want to minimize false negatives further while allowing false positives in a limited level. We believe that our approach is worthy of investigation even though after all we may not collect some web pages due to false positives.

We define two metrics: the redundancy rate and the coverage loss rate. The redundancy rate shows how many web pages are duplicated due to false negatives. The coverage loss rate shows how many web pages are lost due to allowing false positives. Over 170 million URLs that were collected in the real web sites in Korea, we report statistical information on the redundancy and loss of web pages for each method of the URL normalization. The investigation shows the cases in which our approach works well in practice. Analyzing the statistical data on the effect of the URL normalization, we propose the new methods for URL normalization that reduces the rate of false negatives significantly while allowing false positives just a little.

Our paper is organized as follows. In section 2, the syntax of a URL and the standard URL normalization are presented. Section 3 describes the two metrics, and section 4 presents our transformation options for the URL normalization along with experimental results. Section 5 contains the closing remarks.

## 2 URLs and Their Normalization

A URL is composed of five components: the scheme, authority, path, query, and fragment components. The scheme component contains a protocol (here, Hypertext Transfer Protocol) that is used for communicating between a web server and a client. The authority component has three subcomponents: user information, host, port. The user information may consist of a user name and, optionally, scheme-specific information about how to gain authorization to access the resource. The user information, if present, is followed by a commercial at-sign (“@”) that delimits it from the host. The host component contains a location of a web server. The location can be described as either a domain name or IP (Internet Protocol) address. A port number can be specified in the component. The colon symbol (“:”) should be prefixed prior to the port number. The path component contains directories including a web page and a file name of the page. A directory and a file are separated by the slash symbol (“/”). The query component contains parameter names and values that may be supplied to web applications. The query string starts with the question symbol (“?”). A parameter name and a parameter value are separated by the equal symbol (“=”). A pair of parameter name and value is separated each other by the ampersand symbol (“&”). The fragment component is used for indicating a particular part of a document. The fragment string starts with the sharp symbol (“#”). Fig. 1 shows all the components of a URL.



**Fig. 1.** URL Example

The URL normalization is a process that transforms a URL into a canonical form. During the URL normalization, syntactically different URLs that are equivalent should be transformed into a syntactically identical URL, and URLs that are not equivalent should not be transformed into a syntactically identical URL. The standard document [1] describes the syntax-based standard URL normalization as below.

First, a URL with a default port number (80 for the HTTP protocol) and a URL without the port number represent the same page. For instance, “http://example.com:80/” and “http://example.com/” represent the same page. During the normalization, the default port number is truncated from a URL.

Second, the scheme and authority components are case insensitive. For example, “http://EXAMPLE.com” and “http://example.com” represent the same page. During the normalization, all the letters in the components are changed into lower-case letters.

Third, a URL with path string null and a URL with path string “/” represent the same page. For instance, “http://example.com” and “http://example.com/” represent the same page. If a path string is null during the normalization, then the path string is transformed into “/”.

Fourth, unreserved characters (namely, uppercase and lowercase letters, decimal digits, hyphen, period, underscore, and tilde) can be encoded into a three-digit string, where the percent symbol (“%”) should be located at the first position, and the last two digits are a hexadecimal number representing an ASCII code of the character under consideration. For instance, “http://example.com/~smith” and “http://example.com/%7Esmith” represent the same page. During the normalization, encoded unreserved characters are decoded.

Fifth, a URL with the fragment and a URL without the fragment represent the same page. For instance, “http://example.com/list.htm#chap1” and “http://example.com/list.htm” represent the same page. During the normalization, the fragment in the URL is truncated.

### 3 Two Metrics

This section presents two metrics (namely the redundancy rate and the coverage loss rate) to analyze the effect of the URL normalization. These metrics will be used to evaluate a normalization method later.

A set of equivalent URL candidates is defined as a set of URLs that may possibly represent the same web page under a particular normalization method. The set of all URLs,  $U$ , is then represented as  $U_1 \cup U_2 \dots \cup U_n$ , where  $U_i$  denotes the  $i$ th set of equivalent URL candidates and  $U_i \cap U_j = \emptyset$  for  $i \neq j$ . It should be noted that the decomposition of  $U$  into  $U_i$  is fully dependent on a normalization method in question. Further, we do not need to consider a set of equivalent URL candidates in which the number of elements is exactly one, because there is no equivalent URL candidate to compare. For the purpose of computation of the redundancy rate and coverage loss rate, we drop off all  $U_i$ 's with only one element without loss of generality.

Suppose that we have  $N$  sets of equivalent URL candidates in which the number of elements is at least two.  $D_i$  is the set of successfully downloaded documents (or pages) from all the URLs in  $U_i$ .  $n(D_i)$  is the number of documents in  $D_i$ ,  $un(D_i)$  is the number of unique documents in  $D_i$  (simply, we count the number of unique contents in  $D_i$ ). The redundancy rate is defined as equation (1).

$$\text{Redundancy rate} = \frac{\sum_{i=1}^N (n(D_i) - un(D_i))}{\sum_{i=1}^N n(D_i)} \quad (1)$$

The redundancy rate shows how many web pages are duplicated due to equivalent URLs. If all the equivalent URL candidates in  $U$  represent different web pages, then the redundancy rate becomes 0.

Suppose that we transform all the URLs in  $U_i$  into a syntactically identical URL, hoping that all the URLs in  $U_i$  are equivalent. During the transformation, a normalization method under consideration could transform a non-equivalent URL into a syntactically identical one (i.e., false positive occurs). This implies that some valid web pages may be lost (not collected) during the transformation. For example, if  $U_i$  is composed of a number of URLs representing different web pages, then  $(un(D_i) - 1)$

documents will be lost. The coverage loss rate is defined in equation (2), where  $D_{\text{downloaded}}$  denotes the number of documents downloaded successfully with the transformed URLs.

$$\text{Coverage loss rate} = 1 - \frac{D_{\text{downloaded}}}{\sum_{i=1}^N un(D_i)} \tag{2}$$

In essence, the redundancy rate shows how much web pages are redundantly collected due to false negatives, and the coverage loss rate shows how much valid web pages are lost due to false positives. Our approach toward the URL normalization is to reduce the redundancy rate significantly while limiting the coverage loss rate to be within a (user-definable) threshold.

**Example 1.**

Assume that we have 10 URLs (u1 to u10) as below. For convenience, the scheme component (i.e. “http://”) is omitted.

$U = \{ \text{www.microsoft.com/ (u1), www.microsoft.com/default.asp (u2), } \\ \text{www.microsoft.com/index.htm (u3), www.microsoft.com/index.html (u4), } \\ \text{nasdaq.com/asp/ownership.asp (u5), nasdaq.com/ASP/ownership.asp (u6), } \\ \text{ietf.org/tao.html (u7), ietf.org/TAO.html (u8), ietf.org/Tao.html (u9), } \\ \text{www.acm.org/ (u10) } \}$

We can think of several criteria to decompose U into sets of equivalent URL candidates. Suppose that we decompose U under the designation of default pages of web servers. Then we have seven sets of equivalent URL candidates:

$U_1 = \{u1, u2, u3, u4\}, U_2 = \{u5\}, U_3 = \{u6\}, U_4 = \{u7\}, \\ U_5 = \{u8\}, U_6 = \{u9\}, U_7 = \{u10\}.$

The sets  $U_2, \dots, U_7$  have only one member, so those sets are not considered any more. Suppose that the downloaded pages from URLs u1, u2, u3, and u4 are  $\text{\textcircled{a}}$ ,  $\text{\textcircled{a}}$ ,  $\text{\textcircled{b}}$ , and  $\text{\textcircled{b}}$ , respectively. Assume that u1 is the normalized URL for  $U_1$  (i.e., u2, u3 and u4 are transformed to u1). Then we have following computation:

The redundancy rate =  $(4 - 2)/4 = 0.5$ ,

The coverage loss rate =  $1 - (1/2) = 0.5$  (note that page  $\text{\textcircled{b}}$  is lost).

Now consider that we decompose U under other criterion (such as the case sensitivity at the path component). Then we have seven sets of equivalent URL candidates:

$U_1 = \{u1\}, U_2 = \{u2\}, U_3 = \{u3\}, U_4 = \{u4\}, \\ U_5 = \{u5, u6\}, U_6 = \{u7, u8, u9\}, U_7 = \{u10\}.$

Suppose that the downloaded pages from URLs u5, u6, u7, u8, and u9 are  $\text{\textcircled{c}}$ ,  $\text{\textcircled{c}}$ ,  $\text{\textcircled{d}}$ ,  $\text{\bullet}$ , and  $\text{\bullet}$ , respectively, where  $\text{\bullet}$  denotes the downloading failure. Assume that u5 and u7 are the normalized URLs for  $U_5$  and  $U_6$ , respectively. Then we have following computation:

The redundancy rate =  $((2 - 1) + (1 - 1)) / (2+1) = 0.33$ ,

The coverage loss rate =  $1 - (2/2) = 0$ .

It should be noted that the set of all URLs can be decomposed differently, depending on the decomposition criteria. ■

## 4 New URL Normalization Methods

This section considers additional normalization steps beyond the ones specified in the standard document [1]. In our experiment, the web robot [4] crawled approximately the 350,000 Korean sites in the breadth-first fashion during one week in December 2003. The robot requested web pages within nine hops from the root pages of a site. The robot was not allowed to download dynamically-generated pages, whose URLs contain the question symbol inside. Consequently, 50 million web pages were downloaded successfully. We extracted 170 million URLs from the collected web pages, and normalized the URLs according to the standard URL normalization. Using the real URLs we collected, we did an experiment to get the effectiveness of normalization options for URLs in each case.

### 4.1 Case Sensitivity at the Path Component

The Windows operating system manages names of directories and files in a case-insensitive fashion. In a web server working on the Windows operating system, URLs can be composed with various combinations of the upper-case and lower-case letters. For instance, “<http://www.nasdaq.com/asp/ownership.asp>” and “<http://www.nasdaq.com/ASP/ownership.asp>” are equivalent. On the other hands, since the Unix and Linux operating systems manage names of directories and files in a case-sensitive fashion, two URLs composed with different combinations of the upper-case and lower-case letters are very likely to represent different web pages. For instance, “<http://www.acm.org/pubs/journals.html>” and “<http://www.acm.org/PUBS/journals.html>” should be considered not to represent the same web page.

The scheme and host components of a URL are defined to be case-insensitive. The path component of a URL is case-sensitive in principle. Here, we investigate the effects of assuming that the path component is case-insensitive. The set of equivalent URL candidates contains URLs that are syntactically identical except the case sensitivity in the path component. We consider the following three transformation options for the equivalent URL candidates.

- (1) Keep a URL as it is
- (2) Change letters in the path component into the lower-case letters
- (3) Change letters in the path component into the upper-case letters

The first option is the one accepted in the standard URL normalization. The second and third options can cause false positives to occur, because the file and directory names are case-sensitive in the Unix-like operating systems. Analyzing statistical data in our experiment, this paper explores the possibility of accepting the second or third options in the URL normalization.

In our experimental set of URLs, there were 383,444 equivalent URL candidates, which were decomposed into 185,474 sets of equivalent URL candidates. Figure 2 shows the number of requested, downloaded, and unique pages for each option. In the first option, the redundancy rate was 0.52. That is, over half of the successfully downloaded web pages were turned out duplicates.

In the second option, we change all the letters in the path component into the lower-case letters, resulting that each set of equivalent URL candidates has only one URL after normalization. Hence, only one request for the each set is possible. In terms of the number of web requests, there was 48% (= 185,474 / 383,444) reduction after normalization. The coverage loss rate of the second option was 0.01, which means 1% of the contents were lost. As for the third option, the coverage loss rate was 0.11.

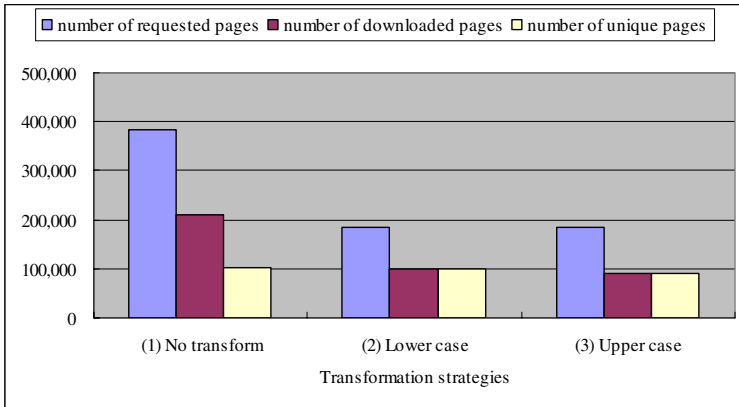


Fig. 2. Results for case sensitivity at the component

In terms of case sensitivity at the path component of a URL, almost all equivalent URL candidates in the same set represent the same page in the real web. We could download more web pages in the second option than those in the third option. In contrast to the first option, the second option (transforming all the letters in the path component into the lower-case letters) not only reduced the number of requests by 48% but also reduced redundant web pages by 52% at the cost of losing 1% of the unique pages that should be downloaded. Taking into account the benefits of the second option, we would say that the coverage loss rate (here, 1%) is negligible.

#### 4.2 The Last Slash Symbol at the Non-empty Path Component

The standard URL normalization defines an empty path of a URL to be equivalent to “/”, thus the two URLs are equivalent: “http://example.com” and “http://example.com/”. The standard URL normalization, however, does not do anything on the last slash symbol at the non-empty path component at all (i.e., leaves it as it is).

This subsection considers the last slash symbol at the non-empty path component of URL. A URL with the last slash symbol represents a directory. When sending web servers such URL, web clients get either a default page in the requested directory or a temporarily created page showing all files in the directory. Users requesting a directory often omit specifying the last slash symbol in a URL. What really happens in this case is that web servers are likely to redirect the URL into a URL including the

last slash symbol. For instance, “http://acm.org/pubs” is redirected into “http://acm.org/pubs/”. This redirection allows users not to take care of the last slash symbol when users specify URLs.

Two URLs with and without the last slash symbol in the non-empty path component consist of a set of equivalent URL candidates. Each set of equivalent URL candidates has exactly two URLs that differ only at the last slash symbol. We consider the following three transformation options for the equivalent URL candidates.

- (1) Keep a URL as it is
- (2) Pick the URL with the last slash symbol as a normalized URL.
- (3) Pick the URL without the last slash symbol as a normalized URL.

In our experimental set of URLs, there were 152,924 equivalent URL candidates, which were decomposed into 76,462 sets of equivalent URL candidates. Figure 3 shows the number of requested, downloaded, and unique pages for each option. In the first option, the redundancy rate was 0.49, which means approximately half of the successfully downloaded web pages were duplicates. The coverage loss rates in the second and third options were 0.01 and 0.03, respectively.

Our experiment shows that the possibility that equivalent URL candidates represent different pages is indeed rare. In the second and third options, the coverage loss rates were very small (0.01 and 0.03, respectively). The second option (specifying the last slash symbol at the end of the URL if possible) could reduce 50% redundant web pages, while only losing 1% of the unique pages to be downloaded.

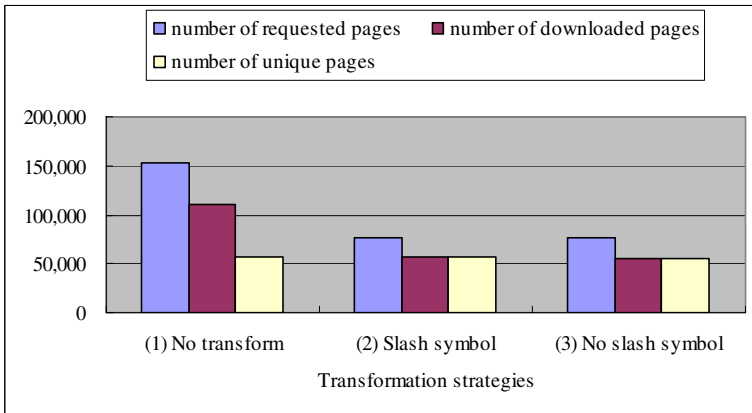


Fig. 3. Results for the last slash symbol

### 4.3 The Designation of a Default Page

A default page is a file to look for, when a client requests a directory. The default page can be specified in a URL. For instance, “http://www.acm.org/” and “http://www.acm.org/index.htm” represent the same page in the site (acm.org), which

sets the designation of a default page as “index.htm”. This subsection considers the designation of default pages in a web site.

In reality, any file name could be designated as a default page. It is reported [6] that only two web servers (the Apache web server, the MS IIS (Internet Information Services) web server) comprise over 85% of all the installed web servers in the world. The default pages of those web servers are “index.htm”, “index.html”, or “default.htm”, when they are installed by default. We consider the three file names, i.e., “index.htm”, “index.html”, and “default.htm”, as default pages in our experiment.

The URLs with and without the default page names consist of a set of equivalent URL candidates. We consider the following two transformation options for the equivalent URL candidates.

- (1) Keep a URL as it is.
- (2) Eliminate default page names in the path component. This means that a URL with a default page specified in the path component is normalized by eliminating the default page name.

In our experimental set of URLs, there were 147,266 sets of equivalent URL candidates and 305,835 equivalent URL candidates. Figure 5 shows the number of requested, downloaded, and unique pages for each option. The redundancy rate of the first option was 0.42. The coverage loss rate of the second option was 0.19. The second option could reduce 42% of redundant pages, but lost 19% of the unique web pages to be downloaded.

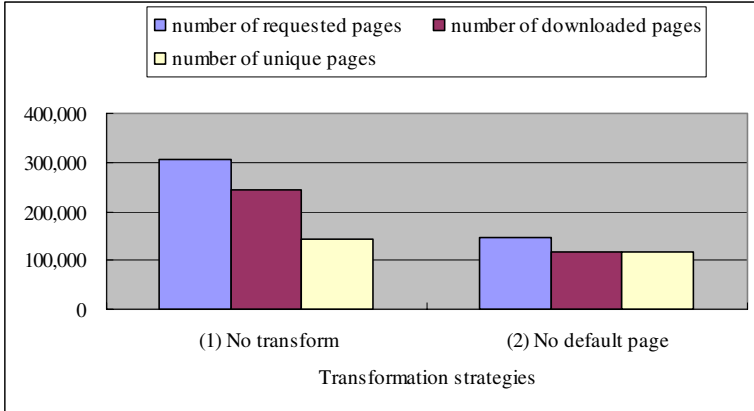


Fig. 4. Results for the designation of a default page

## 5 Conclusion and Future Work

We have considered three additional normalization steps beyond the ones specified in the standard document. Using two metrics this paper defined, we have analyzed the effects of each normalization step on the set of real URLs we collected in the all the Korean web pages. Our experimental results are summarized in Table 1. Changing all

letters in the path component into low-case letters and specifying the last slash symbol could reduce approximately 50% redundant web pages with the small coverage loss rates (i.e., 0.01). Despite the reductions of the redundancy rates, eliminating a default page name caused a large portion (19%) of the web pages to be lost.

**Table 1.** Summary of experimental results

| Type                                    | Normalization options         | Redundancy rate | Coverage loss rate |
|---|-------------------------------|-----------------|--------------------|
| Case sensitivity at the path component  | Leave as it is                | 0.52            | 0                  |
|   | Change into lower cases       | 0               | 0.01               |
|   | Change into upper case        | 0               | 0.11               |
| Last slash symbol at the path component | Leave as it is                | 0.49            | 0                  |
|   | Slash symbol                  | 0               | 0.01               |
|   | No slash symbol               | 0               | 0.03               |
| Default pages                           | Leave as it is                | 0.42            | 0                  |
|   | No page default specification | 0               | 0.19               |

Under the two cases (case sensitivity at the path component and last slash symbol), does work out the idea that we can reduce the redundant web pages significantly at the cost of allowing false positives at a limited threshold. Adoption of our approach, which allows false positives to take place as well as reduces false negatives considerably, is completely up to the users. If efficiency is more important than correctness in URL process of web applications, our approach is an excellent choice. But if users think that correctness is more important than efficiency, they will stick to only the standard URL normalization.

The URL normalization is composed of the several steps to transform a URL into another. The order of transformation steps could lead to different URLs. As a future work, we plan to investigate how to effectively establish the normalization steps that consider not only the steps in the standard community but also the steps described in this paper.

## References

1. Berners-Lee, T., Fielding, R., and Masinter, L.: Uniform Resource Identifiers (URI): Generic Syntax, <http://gbiv.com/protocols/uri/rev-2002/rfc2396bis.html> (2004)
2. Burner, M.: Crawling Towards Eternity: Building an Archive of the World Wide Web, *Web Techniques Magazine*, Vol. 2, No. 5. (1997) 37-40
3. Heydon, A. and Najork, M.: Mercator: A Scalable, Extensible Web Crawler, *International Journal of WWW*, Vol. 2, No. 4. (1999) 219-229
4. Kim, S.J. and Lee, S.H.: Implementation of a Web Robot and Statistics on the Korean Web, *Springer-Verlag Lecture Notes in Computer Science*, Vol. 2713, (2003) 341-350
5. Kim, S.J. and Lee, S.H.: How Web Pages Change: An Empirical Study, submitted for publications (2004)