Mirror Site Maintenance Based on Evolution Associations of Web Directories

Ling Chen
L3S/University of Hannover
Appelstr. 9a
30167 Hannover, Germany
lchen@l3s.de

Sourav Bhowmick
SCE, Nanyang Technological University
Singapore, 639798
assourav@ntu.edu.sg

Wolfgang Nejdl
L3S/University of Hannover
Appelstr. 9a
30167 Hannover, Germany
nejdl@l3s.de

ABSTRACT

Mirroring Web sites is a well-known technique commonly used in the Web community. A mirror site should be updated frequently to ensure that it reflects the content of the original site. Existing mirroring tools apply *page-level* strategies to check each page of a site, which is inefficient and expensive. In this paper, we propose a novel *site-level* mirror maintenance strategy. Our approach studies the evolution of Web directory structures and mines association rules between ancestor-descendant Web directories. Discovered rules indicate the evolution correlations between Web directories. Thus, when maintaining the mirror of a Web site (directory), we can optimally skip subdirectories which are negatively correlated with it in undergoing significant changes. The preliminary experimental results show that our approach improves the efficiency of the mirror maintenance process significantly while sacrificing slightly in keeping the "freshness" of the mirrors.

Categories and Subject Descriptors: H.2.8 [Database Management]: Database Applications[data mining]

General Terms: Measurement, Performance, Experimentation.

Keywords: Web evolution, evolution correlation, mirror maintenance.

1. INTRODUCTION

A mirror site is a replica of an original Web site and is often located in a different place throughout the Internet. There are lots of reasons for Web mirroring such as load reduction, better availability, and faster access etc [3]. Web mirroring can be generally classified as internal mirroring and external mirroring. The former refers to the situation that both the original site and mirror sites are set up and maintained by the same organization, such as *www.akamai.com*. The latter means that the original site and mirror sites are autonomous. For example, many Linux user groups in different countries, such as *www.linuxforum.net* in China, mirror the site *www.samba.org* to provide quick access for Linux fans in their own countries.

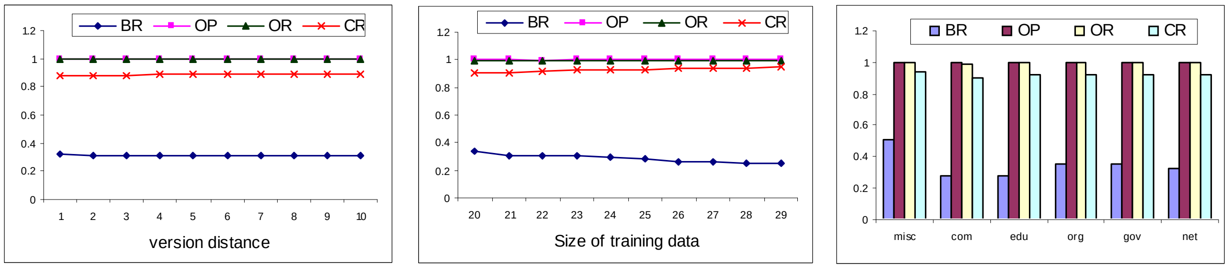
A mirror site should be updated regularly with respect to the original site. For internal mirroring, this process can be performed efficiently by synchronizing mirror sites only if the original site evolves. However, for external mirroring, the server holding the mirror site has to scan the original site frequently to detect the changes. To maintain the freshness of mirror sites, *www.samba.org* suggests external mirror sites scan it once per day. Obviously, daily scan of the whole original site is not a light workload for the server holding the mirror site, especially when the server holds more than one mirror sites. Hence, optimized mirror maintenance strategies are needed so that mirroring severs do not need to scan the whole original site every day.

In this paper, we propose a site-level mirror maintenance strategy based on the historical evolution of the original Web site. Since existing Web mirroring tools, like "rsync" [1], usually mirror a site according to its Web site directory tree, we study the evolutionary characteristics of Web site directory structure. Particularly, we discover *Negative Evolution Association Rules* (*NEAR*) of ancestor-descendant subdirectories from the evolution history of the site. A *NEAR* $s_a s_b \cdots s_k \Rightarrow \neg s_{k+1}$, where s_a through s_{k+1} is a sequence of Web directories with ancestor-descendant relationships, indicates that $s_a s_b \cdots\ s_k$ frequently change together while s_{k+1} rarely undergoes significant changes together with its ancestor directories $s_a s_b \cdots\ s_k$. Based on discovered rules, when maintaining the mirror of a changed target directory (e.g., s_a), its subdirectories which frequently change together with it (e.g., $s_b \cdots s_k$) should be maintained together. However, the mirror maintenance process can be optimized by skipping subdirectories which have negative evolution associations with it (e.g., s_{k+1}).

2. NEAR MINING

The file system of a Web site can be represented as a directory tree $S = \langle N, E\rangle$, where N is the set of nodes corresponding to files or directories in the server, E is the set of edges representing the consisting-of relationship between corresponding directories and files. Accordingly, each Web subdirectory can be represented as a subtree. A directory subtree s_a is an ancestor of another subtree s_b, denoted as $s_a \succ s_b$, if there is a path from the root of s_a to the root of s_b. Let $\mathcal{T} = \langle t_1, t_2, \ldots, t_n\rangle$ be a sequence of time points with a particular time granularity. Given a directory tree S of some original Web site, we study the sequence of historical versions of S on $\mathcal{T}$, which is denoted as $\langle S^{t_1}, S^{t_2}, \ldots, S^{t_n}\rangle$.

Before defining *NEARs*, we first define three evolution metrics, *Degree of Change*, *Frequency of Change* and *Correlation of Change*. Given two versions of a tree structure, S^{t_i} and $S^{t_{i+1}}$, let $D(S^{t_i}, S^{t_{i+1}})$ be the edit distance [5] between the two versions. The *Degree of Change* of the tree, denoted as $DoC(S^{t_i}, S^{t_{i+1}})$, is $\frac{D(S^{t_i}, S^{t_{i+1}})}{|S^{t_i} \uplus S^{t_{i+1}}|}$, where $|S^{t_i} \uplus S^{t_{i+1}}|$ is the



(a) variation of distance

(b) variation of training data size

(c) performance on domains

Figure 1: Performance of Mirror Maintenance.

size of the merged tree of S^{t_i} and $S^{t_{i+1}}$. The value of DoC ranges from 0 to 1. The greater the value of DoC, the more significantly the tree changes.

Given a sequence of tree versions $\langle S^{t_1}, S^{t_2}, \ldots, S^{t_n}\rangle$, the *Frequency of Change* of a set of subtrees $X = \{s_1, s_2, \ldots, s_m\}$, with respect to some DoC threshold α, denoted as $FoC_\alpha(X)$, is $\frac{\sum_{j=1}^{n-1} G_j}{n-1}$, *where* $G_j = \prod_{i=1}^{m} G_{j_i}$ and $G_{j_i} = \begin{cases} 1, & \text{if } DoC(s_i^{t_j}, s_i^{t_{j+1}}) \geq \alpha \\ 0, & \text{otherwise} \end{cases}$. That is, FoC_α of a set of subtrees is the fraction of versions where the set of subtrees undergo significant changes together. The greater the value of FoC, the more frequently the set of subtrees change together. We adopt the ϕ-coefficient to define the *Correlation of Change* between two subtree sets. The CoC between two subtree sets X and Y ($X \cap Y = \emptyset$), with respect to some DoC threshold α, denoted as $CoC_\alpha(X,Y)$, is $\frac{FoC_\alpha(X \cup Y) - FoC_\alpha(X) * FoC_\alpha(Y)}{\sqrt{FoC_\alpha(X)(1-FoC_\alpha(X))FoC_\alpha(Y)(1-FoC_\alpha(Y))}}$. According to the definition of ϕ-coefficient, if $CoC_\alpha(X,Y)$ is greater than zero, X and Y are positively correlated in undergoing significant changes. Otherwise, they are negatively correlated.

Based on the evolution metrics described above, *Negative Evolution Patterns* can be defined as follows. Given DoC threshold α, FoC threshold β, and CoC threshold γ ($0 \leq \alpha, \beta \leq 1, \gamma \geq 0$), $P = \langle X, Y\rangle$, where $X = \langle s_1, s_2, \ldots, s_k\rangle$, $Y = \langle s_{k+1}\rangle$, and $s_i \succ s_{i+1}$ ($1 \leq i \leq k$), is a *Negative Evolution Pattern* (NEP) if (i) $FoC_\alpha(X) \geq \beta$; (ii) $FoC_\alpha(X \cup Y) < \beta$; ($iii$) $CoC_\alpha(X,Y) \leq -\gamma$. Given a NEP $\langle\langle s_1 s_2 \cdots s_k\rangle \langle s_{k+1}\rangle\rangle$, valid $NEAR$ $s_1 s_2 \cdots\ s_k \Rightarrow \neg s_{k+1}$ can be derived if its *confidence* is no less than some specified threshold. Similar to traditional association rule mining, the *confidence* of the $NEAR$ can be computed as $\frac{FoC(X) - FoC(X \cup Y)}{FoC(X)}$.

2.1 Overview of Algorithm

Given a sequence of historical tree versions and the set of related thresholds, the mining of $NEARs$ can be decomposed into the following two phases.

Phase I: Historical change organization. In this phase, we first use an algorithm WD-Diff, which modifies the existing algorithm X-Diff [5], to efficiently detect changes between each two successive Web directory tree versions. We then organize detected changes into a special data structure, called *General Delta Tree* (GDT), which not only records the change information of subtrees but also preserves the ancestor-descendant relationships between subtrees.

Phase II: Rule generation. We discover $NEARs$ from the GDT with respect to the set of given thresholds. A divide-and-conquer strategy is employed to discover $NEARs$ starting with different Web directory subtrees. The details of the algorithm can be referred to [2].

Since most of existing mirroring tools offer the option which allows to skip mirroring a particular subdirectory, discovered $NEARs$ can be used together with existing tools. Basically, when performing a top-down traversal on the target directory tree, our optimized mirror maintenance approach either skips a subdirectory based on $NEARs$ or mirrors pages using existing tools. The details of proposed mirror maintenance approach can be referred to [2].

3. EXPERIMENTAL RESULTS

We evaluate the performance of our optimized mirror maintenance approach based on real data collected by A. Ntoulas et al. [4], who downloaded pages from 154 "popular" Web sites every week from October 2002 until October 2003, for a total of 51 weeks. The following three experiments are conducted and the respective results are reported in Figure 1 (a), (b) and (c). (i) $NEARs$ mined from the first 20 versions are used to maintain mirrors and the quality of the maintained version is evaluated with respect to the $(20+k)th$ ($k \geq 1$) version. We vary the value of k to study the temporal validity of discovered rules. Four evaluation metrics are used: *Bypass Ratio* (BR), *Overall Precision* (OP), *Overall Recall* (OR), and *Change Recall* (CR). *Bypass Ratio* is the fraction of Web pages skipped by our approach. *Overall Precision* and *Overall Recall* measure the accuracy of our approach with respect to the whole test version. *Change Recall* measures the accuracy of our approach with respect to changes to the test version. As a good mirror maintenance approach, values of all the four metrics should be as high as possible. We observed that generally, our approach skips around one third pages of a site and loses slight "freshness". The performance does not deteriorate with the increase of k. (ii) $NEARs$ are mined from the first k versions and the subsequent version is used as the test data. The results show that accuracy increases while efficiency decreases slightly. (iii) We evaluate the effectiveness of our approach on sites from different domains, such as *.com*, *.gov*, *.org* etc. The results indicate our approach is applicable to sites from different domains.

4. CONCLUSIONS

In this paper, we proposed an optimized mirror maintenance approach based on $NEARs$ discovered from the evolution history of a Web site. Promising experimental results showed the effectiveness of this approach.

5. REFERENCES

[1] Rsync. In *http://samba.anu.edu.au/rsync/*.
[2] Web mirroring project. In *http://www.l3s.de/~lchen/mirror.pdf*.
[3] K. Bharat and A. Z. Broder. Mirror, mirror on the web: A study of host pairs with replicated content. *Computer Networks*, 1999.
[4] A. Ntoulas, J. Cho, and C. Olston. What's new on the web?: the evolution of the web from a search engine perspective. In *WWW*, 2004.
[5] Y. Wang, D. J. DeWitt, and J.-Y. Cai. X-diff: An effective change detection algorithm for xml documents.. In *ICDE*, 2003.