A Hybrid Approach for Refreshing Web Page Repositories

M. Ghodsi, O. Hassanzadeh, Sh. Kamali, and M. Monemizadeh
{ghodsi, hassanzadeh, kamali, monemi}@ce.sharif.edu

Computer Engineering Department
Sharif University of Technology, Tehran, Iran

Abstract. Web pages change frequently and thus crawlers have to download them often. Various policies have been proposed for refreshing local copies of web pages. In this paper, we introduce a new sampling method that excels over other change detection methods in experiment. Change Frequency (CF) is a method that predicts the change frequency of the pages and, in the long run, achieves an optimal efficiency in comparison with the sampling method. Here, we propose a new hybrid method that is a combination of our new sampling approach and CF and show how our hybrid method improves the efficiency of change detection.

Keywords: web crawling, change detection, sampling, change frequency, hybrid algorithm

1 Introduction

Crawlers and spiders download and accumulate web pages to be categorized and ranked fast by search engines in response to users' queries. On the other side, web pages are updated frequently; new pages appear and old pages disappear quite often. To refresh web repositories, crawlers have to download new pages, clean the page archive from disappeared pages and update it with the new ones.

In [4] *Change Frequency (CF) method* was introduced for estimating the frequency of changes in web pages. Based on the past change history of the pages, this method estimates how often a page changes and decides on how often it must be revisited.

Sampling method is another approach for downloading web pages [5]. In this method, a small number of pages from each web site is sampled and the number of changed pages is then estimated. Based on the estimates, the download resource for each web site is allocated accordingly.

In this paper, we propose a hybrid model that commences to sampling, in the beginning, based on an improved sampling algorithm. With the passage of time, when the sampling resources reduce and when it gains a complete history of the page changes, our algorithm switches to CF and allocates more resources to this method.

In the remainder of the paper, in Section 2, we present our new sampling algorithm. Section 2 describes the CF method followed by the overall hybrid

L. Zhou, B.C. Ooi, and X. Meng (Eds.): DASFAA 2005, LNCS 3453, pp. 588–593, 2005.

method which is proposed in Section 3. Our experimental results are presented in Section 5.

2 Improved Sampling Method

We suppose that all pages are equal in size and the resource available for crawling is counted based on the maximum number of pages we can download. We also assume that there are L web sites each having the same number of pages, denoted by M. The problem is to decide how we can best refresh the repository. Our measure for the optimality of an algorithm is its *efficiency* which is measured by the fraction of total number of downloaded items that have been changed.

Our iterative sampling method proceeds as follows. In each iteration, a fraction R pages (called the *sampling size*) of each site is downloaded uniformly at random. We estimate ρ_i, the percentage of the downloaded pages of site i that have been changed.

Let $P(\rho_i)$ be the probability of downloading another R pages from each site i. We continue the above iteration and download further R pages from site i with this probability.

We propose the following estimate for $P(\rho_i)$ to be calculated in each iteration.

$$P(\rho_i) = \frac{tanh(10(\rho_i - \rho_0) + 1)}{2} \tag{1}$$

where ρ_0 is the point of inflection. Reducing the resources available for sampling is done by increasing the value of ρ_0.

Here we justify the choice of equation 1. We would like to choose values for $P(\rho_i)$ that maximizes the number of detected changed pages in site i and considering our limited resource. For this, we calculate N_i, the average number of detected changed pages in site i, as follows.

We first assume that ρ_i is equal in all iteration. In other words, we assume the estimation of ρ_i is without error; we relax this assumption later. N_i is then computed as follows:

$$\begin{aligned} N_i &= R\rho_i + R\rho_i \sum_{t=2}^{k} tP^{t-1}(\rho_i)(1 - P(\rho_i)) \\ &= R\rho_i * [1 - P^k(\rho_i) + \frac{P(\rho_i)}{1 - P(\rho_i)}(P^{k-1}(\rho_i) - 1) - (k-1)P^k(\rho_i) + P(\rho_i)] \end{aligned}$$

where k is the maximum number of iterations which is equal to $\frac{M}{R}$.

Let N be the total number of detected changed pages. Then,

$$N = R * \sum_{i=1}^{L} \rho_i * [1 - P^k(\rho_i) + \frac{P(\rho_i)}{1 - P(\rho_i)}(P^{k-1}(\rho_i) - 1) - (k-1)P^k(\rho_i) + P(\rho_i)] \tag{2}$$

The best choice for $P(\rho)$ that maximizes the expression in equation 2, taking into account the limited number of pages we can download, is a pulse function (Figure 1.1).

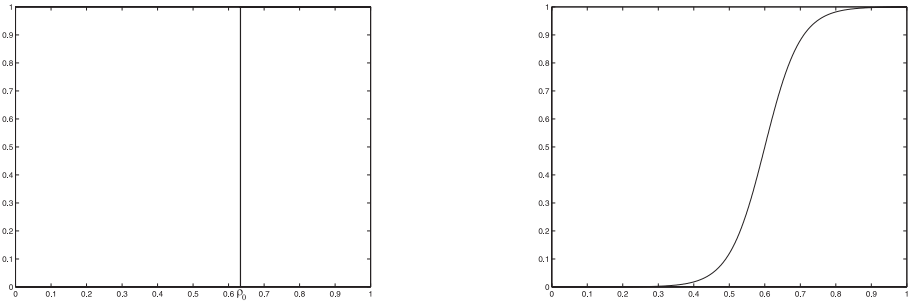

Fig. 1. (1) Diagram of $P(\rho)$ if ρ is estimated without error, (2) Diagram of $P(\rho)$.

The pulse function is the best choice when we can estimate the percentage of changed pages in the site without error. But in practice, this estimation is not exact and contains error. This error can affect the efficiency of our algorithm specially near ρ_0. To remedy this effect we can smooth the diagram of $P(\rho_i)$ near this point. The result is shown in figure 1.2. The diagram of figure 1.2 is very similar to the diagram of equation 2. This justifies why we choose tanh function to represent $P(\rho_i)$.

3 Change Frequency Method

Here, we briefly describe the CF phase. To help our discussion, we first borrow some notation from [4]. We assume n is the number of accesses to the page in interval I and X is the total number of changes detected within these accesses. Let f be the frequency at which we access the element and its value is $f = \frac{1}{I}$. λ, the change frequency of a page, is the number of times that the page was changed during a specific time. Now we can define the *frequency ratio* $r = \frac{\lambda}{f}$, the ratio of the change frequency to the access frequency. We can estimate frequency ratio $r = \frac{\lambda}{f}$ first and estimate λ indirectly from r. We use the estimator $\hat{r} = -\log(\frac{\bar{X}+0.5}{n+0.5})$, where $\bar{X} = n - X$ is the number of accesses that the element did not change, because it is more useful as shown in [4]. Consider two queues and call them *Normal* and *Slow*. Slow queue contains all pages that change rarely and we guarantee to crawl them in every one month, i.e., $I = 30$. The other pages are placed in Normal queue which are guaranteed to be crawled twice per month, i.e., $I = 15$. Over time, pages commute between Normal and Slow queues based on their estimated change frequency.

After creation of Normal and Slow queues, the algorithm works as follows: We crawl pages approximately twice per month in Normal queue and once per month in Slow queue. For each page p in each queue, we download p and compare p with the local copy to determine if it has changed, then calculate the new change frequency for it. If the new change frequency of page was still greater than some

adaptive threshold, we append the page to the end of Normal queue, otherwise the page falls down into Slow queue.

4 The Hybrid Algorithm: An Overall View

4.1 Definitions

We denote a *cycle* to be an interval consisting of a *sampling phase* and a *CF phase*. For each site in a sampling phase a sequence of *step*s are defined where a step is an interval that the algorithm samples a specified number of pages from the site.

4.2 Towards Change Frequency

Sampling performs well without knowing the change history of pages. In contrast, frequency-based policy suffers from poor performance with insufficient knowledge of change history. This happens at the beginning which we have a poor estimation of change frequencies. But as time passes on, its performance improves over the sampling, as the history of changes converges to its more accurate values. So, it seems that we can gain more efficiency if we allocate more resources to CF as our crawling progresses. The migration towards CF should be done slowly. This is an important part of our algorithm whose overall view is described in the following algorithm.

Algorithm: HybridAlgorithm()
Input:
ρ_0: The inflection point that takes its value from [0..1].

```
1   Crawl all pages in the beginning and store them in repository REP
2   while True
3         do Increase ρ0 one step
4            Call SamplingPhase
5            Call ChangeFrequencyPhase
```

Above algorithm is based on allocating resources between its two phases. We define a function that shows the used resource in the sampling phase. By adjusting the value of this function we can migrate from sampling to CF. For this purpose, we adjust the value of parameter ρ_0. As we increase it, the diagram of P shifts to the right. Let $U(\rho_k)$ be the average number of sampling phases in a cycle that we download from site k.

$$U(\rho_k) = \sum_{i=2}^{\frac{M}{R}} iP(\rho_k)^{i-1}(1 - P(\rho_k)) + 1 \tag{3}$$

The total number of resources that will be used in sampling phase is $\sum_{i=1}^{L} RU(\rho_i)$. Thus, by reducing the probability $P(\rho_i)$, the value of $U(\rho_i)$ will be dwindled and eventually tends to one.

5 Experimental Results

For evaluating the various strategies as discussed above, we conducted a number of experiments. In this section, we first describe the distribution of our data-set and then compare various algorithms from the perspective of efficiency.

5.1 Our Data-Set

We collected over 100,000 pages (1000 web pages in 100 web sites) and kept track of their changes over two months. We crawled every site twice per month. It seems that 4 cycles for tracing the changes is relatively small, but based on the previous works [4, 5], we extend our cycles up to 500 cycles (i.e., we repeated the 4 cycles 125 times without actual downloading). We then simulated adaptive sampling, our new sampling, and hybrid algorithms on our data-set.

5.2 Comparison of Algorithms

As you see in figure 2, adaptive sampling has an efficiency of around 75 percent. Our new sampling, the first algorithm presented in this paper, has efficiency of approximately 81 percent. The other algorithm is hybrid algorithm which is a combination of improved sampling and CF and has the average efficiency of about 87 percent over time. As we observe in the figure, still there is a narrow fluctuation in the diagram of CF.

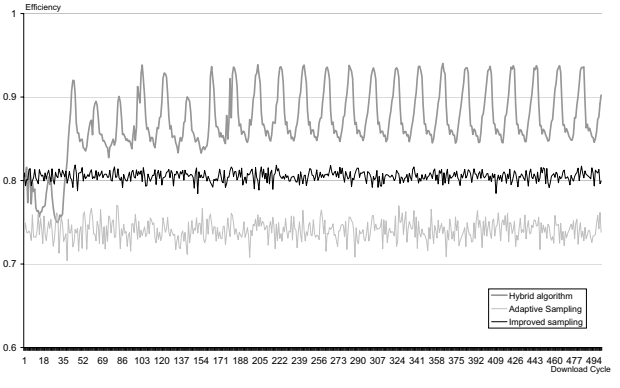

Fig. 2. Comparison of the various download policies over time

6 Conclusion

In this paper, we studied how we can detect changes effectively using a hybrid policy. We introduced a new sampling method that excels over other change detection methods in experiments and suggested improvements to Change

Frequency(CF) in order to increase its efficiency. We also proposed a new hybrid method that is a combination of our new sampling and CF and finally showed how the new method improves efficiency of change detection.

References

1. S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In Proc. WWW conf., April 1998.
2. J. Cho and H. Garcia-Molina. The evolution of the web and implications for an incremental crawler. In Proc. 26th VLDB Conf. , September 2000.
3. J. Cho and H. Garcia-Molina. Synchronizing a database to improve freshness. In Proc. SIGMOD Conf., May 2000.
4. J. Cho. Crawling the web: Discovery and maintenance of a large-scale web data. PhD. Thesis, Stanford University, 2001.
5. A. Ntoulas and J. Cho. Effective change detection using sampling. In Proc. 28th VLDB Conf., Hong Kong, China, 2002.