

# WEB DOCUMENTS PRE-FETCHING BASED ON AN ENSEMBLE OF CLASSIFIERS

A. Georgakis and H. Li\*

Digital Media Laboratory (DML)

Department of Applied Physics and Electronics,

Umeå University, SE-90187, Sweden

apostolos.georgakis@tfe.umu.se.

## ABSTRACT

This paper provides a novel transparent and speculative algorithm for content based web page pre-fetching. The algorithm relies on a profile that is dynamically generated when the user is browsing the Internet and is constantly updated. The objective is to reduce the user perceived latency by predicting future actions. The profile is generated using the adaboost algorithm on the text anchored around the links. By selecting the terms that are highly correlated with the preferences of the user, forthcoming actions can be easily anticipated. The proposed algorithm is tested against an algorithm that relies on the frequency of occurrence for selected terms in the anchored text around the links. A comparison between the proposed algorithm against the tested one yields improved cache-hit rates given a moderate bandwidth overhead. Furthermore, the precision of accurately for the adaboost algorithm is proven to be superior to the frequency-based opponent.

## KEY WORDS

Web prefetching, adaboost algorithm, user profile, future actions.

## 1 Introduction

Due to the growing popularity of the *World Wide Web* (WWW), the subsequent increase of the WWW related network traffic, the size of the web pages as well as the explosive use of multimedia material, for most internet users, retrieval times for documents hosted in many of the web servers are extremely high. The *User Perceived Latency* (UPL) increases further when the user's internet supplier side is added to the above scheme. In that case, downloading times become a product of the connection speeds from both the user's *Internet Service Provider* (ISP) and the backbone provider of the user's ISP. The carrying capacity of the particular backbone provider and the affiliated ISP varies widely with Web traffic volumes, but still, the problem continues to exist.

It is obvious that good response times have positive effect in the user's productivity and satisfaction. This satis-

faction in the case of on-line transactions (*e-commerce*) is crucial for the survival and the prosperity of any company that relies on the Web for selling. It is shown that the "success" of a Web vendor's site is related to the time it takes to load its web pages. But its not only the on-line vendors that benefit from fast connection times, rather any web site that wants to retain and increase its popularity should address sooner or later this issue. A rather old but still useful study conducted in 1999 from Zona Research Inc. provides evidence that the average response varies from less than three seconds for simple pages to about 12 seconds for transactions, and more than 20 seconds for dial-up users [1]. Furthermore, if a Web site takes more than eight seconds to load, the user is much more likely to become frustrated and leave the site. This is an Internet legend which states that "after eight seconds, visitors will abandon the site and select another competitor online destination".

The UPL can be tackled down from the user's side without extra costs through the usage of the browser's cache [2, 3]. In this solution the documents that are more likely to be accessed are *pre-fetched* to the browser's cache. When a request takes place for one of these documents a *cache-hit* occurs and it is not necessary for the client to fetch them over the Internet, thus avoiding some portion of the possible delays.

A crucial issue here is the effective prediction of the next request and subsequently the modeling of the user actions over time. If an algorithm can accurately determine the next document to be requested and pre-fetch it in time then the UPL tends towards zero. Furthermore, there is no bandwidth misuse. A difficulty emerging here is to determine the document that is more probable to be requested. Unfortunately, since the probability of an accurate prediction is almost always below one, more than one files must be pre-fetched at the same time in order to achieve low UPL. In that case some pre-fetched documents may and will never be used which will result in bandwidth overhead and increased load on the web servers.

This paper presents an algorithm that models the Internet users behavior by relying in the textual content anchored around links. The followed links contain text that captivates the user's interest. Therefore, it is justified to assume that the user's interest, which is reflected by the profile, comprises of the text anchored around the followed

---

\*This work was supported by the European Union Research Training Network (RTN) "MUHCI: Multi-modal Human Computer Interaction (HPRN-CT-2000-00111)"

and non-followed links. After the creation of the user profile, the gathered information can be employed by a pre-fetching mechanism that is responsible for determining which outbound links are more probable to be requested. In doing so, when a new page needs to be validated all the outbound links are extracted along with the text anchored around them. The anchored text along with the user’s profile are exploited in computing weights for these links. Subsequently, the algorithm retrieves a portion of the linked documents according to the assigned weights.

In what follows, section 3 covers the proposed pre-fetching algorithms along with the creation of the user’s profiles and section 4 provides the experimental results for the evaluation of the proposed algorithms.

## 2 Related Work

Very useful overviews on caching and pre-fetching techniques can be found in [4]. Data mining techniques have also been used to extract patterns from Web logs [5, 6]. Moreover, the *Prediction-by-Partial-Match* (PPM) has also been employed extensively in [7]. Furthermore, recommendation systems and web usage mining techniques can be used in order to provide hints to a pre-fetching agent over to which links are more probable to be requested [8]. In [9] the server which gets to see requests from several clients makes predictions while individual clients initiate prefetching. A technique for prefetching composite-multimedia rich pages by using partial prefetching is presented in [10]. Discrete Markovian techniques have been applied on the history of Web page references to recognize patterns in the activity of the use [11]. Pirolli and Pitkow in [12] propose the usage of high order Markov Models and  $n$ -gram modeling in order to predict the behaviour and future actions of a user.

It must be noted here that the benefits of any caching scheme are limited when the content of a web server tends to change very frequently. Furthermore there is no benefit when the user is surfing randomly. In that case there is a waste of bandwidth which can be compensated by the fact that the pre-fetched documents, if the client goes through an intermediate cache repository, may improve significantly the performance of other clients that use the same proxy as well. Finally, it must be noted here that the reduced UPL might affect the user’s overall behaviour.

## 3 Pre-fetching algorithm: Some preliminary definitions

Before proceeding to any pre-fetching actions the following series of preprocessing steps are undertaken on the html page currently displayed in the user’s browser:

- Identification of the outbound links and the anchored text around them.

- Removal of links that definitely should not be pre-fetched. That links comprises of pages which contains time critical data and URLs that contains the term “?”. Moreover, pages already in the cache either from an earlier access or pre-fetch are also ignored except when the content of a page needs to be refreshed. Finally, any dynamically generated pages are also excluded.

Let  $\mathcal{W}(t)$  denote the set of web pages that the user have visited until the time instant  $t$ . Each individual web page in the set  $\mathcal{W}(t)$  contains an arbitrary set of outbound links. Let  $\mathcal{L}(t) = \{\mathbf{l}_1, \mathbf{l}_2, \dots, \mathbf{l}_{|\mathcal{L}(t)|}\}$  denote the set of all the outbound links that comprise the web pages in the set  $\mathcal{W}(t)$  where  $|\cdot|$  denotes the cardinality of a set.

For each link and for the text anchored around it the following text processing actions are performed:

- Removal of numbers and punctuation marks. The sole punctuation mark left intact is the full stop. This is done in order to provide a rough sentence delimiter.
- Stopping and stemming is performed using the well-know Porter stemmer [13].

After the text preprocessing, the resulted anchored text contains probably a plethora of stems. In order to identify which of the stems are correlated with the user’s preferences the so-called *low-to-medium* law will be borrowed from the information retrieval community. According to that law, the bigrams whose frequency is low-to-medium are the most informative ones [14, 15]. The above implies that the stems with high frequency are not informative regarding the actual content; they rather help in the formation of the text inside the sentences and therefore can be eliminated. In the case under consideration the high-frequency stems whose cumulative probability equals 10% of the total probability mass are eliminated [16].

Let  $\mathcal{B}(t) = \{b_1, b_2, \dots, b_{|\mathcal{B}(t)|}\}$  denote the set of bigrams that comprise the anchored text around the link until the time instant  $t$  that survived the above thresholding process. Each outbound link,  $\mathbf{l}_i \in \mathcal{L}(t)$  is associated with a set of bigrams drawn from the set  $\mathcal{B}(t)$ . For example in the link  $\mathbf{l}_i$  corresponds the set  $\{b_{(1)}^i, b_{(2)}^i, \dots, b_{(|\mathbf{l}_i|)}^i\}$  where the notion  $(\cdot)$  denotes sampling from a set.

Finally, let  $\mathcal{K} = \{\kappa_{nr} = -1, \kappa_r = 1\}$  denote the label space, where  $\kappa_{nr}$  corresponds to a non-relevant or non-visited outbound link and  $\kappa_r$  corresponds to a relevant link respectively. Each link in  $\mathcal{L}(t)$  is labeled as either relevant or not according to the time the user has spend in the linked page. That is, if the user has “stayed” in the  $i$ th page more than a predefined time threshold  $t_s$  then the page is regarded as important to his preferences and therefore the link that lead to this page is labelled as relevant.

### 3.1 The Adaboost algorithm

Let  $\mathcal{S}(t) = \{(\mathbf{l}_1, \kappa_{(1)}), (\mathbf{l}_2, \kappa_{(2)}), \dots, (\mathbf{l}_{|\mathcal{L}(t)|}, \kappa_{(|\mathcal{L}(t)|)})\}$  denote the sequence of training examples where  $\mathbf{l}_i \in \mathcal{L}(t)$

and label  $\kappa_{(i)} \in \mathcal{K}$ . Let us assume that we are given a set of real-valued functions, called *weak learners*, which accept as input a sequence of training examples from the set  $\mathcal{L}(t)$  and will be denoted by  $h_1, \dots, h_N$  ( $h_j : \mathcal{L}(t) \times \mathcal{K} \rightarrow \mathbb{R}, k \in \{1, 2, \dots, N\}$ ). The adaboost algorithm studies the problem of approximating the  $\kappa_{(i)}$  for each  $\mathbf{l}_i$  using a linear combination of the weak learners (*training phase*). That is, it tries to find a vector of *predictors*  $\mathbf{a} = [a_1, a_2, \dots, a_N] \in \mathbb{R}^N$  such that the function

$$f(\mathbf{l}_i, \kappa_{(i)}) = \sum_{k=1}^N a_k h_k(\mathbf{l}_i, \kappa_{(i)}) \quad (1)$$

to be a ‘‘good approximation’’ of  $\kappa_{(i)}$ . The previous can be interpreted as follows:

$$\text{sign}(f(\mathbf{l}_i, \kappa_{(i)})) = \text{sign}(\kappa_{(i)})$$

or alternatively as:

$$\text{abs}(f(\mathbf{l}_i, \kappa_{(i)})) > \text{abs}(f(\mathbf{l}_i, \mathcal{K} - \{\kappa_{(i)}\})).$$

The absence of equality in the previous expression is explained by the need to be able to differentiate between the two labels in  $\mathcal{K}$ . Furthermore, Eq. (1) can be used in labelling a new and unseen link,  $\mathbf{l}_{new}$ , during the *testing phase*. For example, the sign of the expression

$$\text{sign}(\max(f(\mathbf{l}_{new}, \kappa_{nr}), f(\mathbf{l}_{new}, \kappa_r))) \quad (2)$$

is interpreted as the label to be assigned to  $\mathbf{l}_{new}$ . Moreover, the magnitude:

$$\text{abs}(\max(f(\mathbf{l}_{new}, \kappa_{nr}), f(\mathbf{l}_{new}, \kappa_r))) \quad (3)$$

will be interpreted as the confidence of the prediction for the label given from Eq. (2).

Furthermore, let  $\mathbf{D}_k = \{d_1^k, d_2^k, \dots, d_{|\mathcal{L}(t)|}^k\}$  denote a distribution, where  $d_i^1 = 1/|\mathcal{L}(t)|, i \in \{1, 2, \dots, |\mathcal{L}(t)|\}$ , which corresponds to the importance that the adaboost algorithm places on each instance of the sample space  $\mathcal{S}(t)$ . On each iteration  $k \in \{1, 2, \dots, T\}$  the distribution  $\mathbf{D}_k$  is used to compute a weak hypothesis  $h_k$  over the set  $\mathcal{S}(t)$ . The weak classifier is trained with more emphasis on certain patterns, using a cost function weighted by a probability distribution  $\mathbf{D}_k$  over the training data. The idea behind adaboost is to modify the distribution  $\mathbf{D}_k$  over  $\mathcal{S}(t)$  in a way that will increase the probability mass of the misclassified parts of the domain  $\mathcal{S}(t)$ . In this case sampling with replacement (using the probability distribution  $\mathbf{D}_k$ ) can be used to approximate a weighted cost function. Examples with high probability would then occur more often than those with low probability, while some examples may not occur in the sample at all although their probability is not zero. This will subsequently force the classifier to concentrate on the misclassified training examples and avoid misconceptions in the future.

At the time instance  $k + 1$ , through the usage of the weak hypothesis  $h_k$ , the boosting algorithm generates the

new distribution  $\mathbf{D}_{k+1}$  using the following

$$d_n^{k+1} = \frac{d_n^k \exp\{-a_k \kappa_{(n)} h_k(\mathbf{l}_n, \kappa_{(n)})\}}{Z_{k+1}} \quad (4)$$

where  $Z_{k+1}$  is a normalization factor equal to

$$Z_{k+1} = \sum_{i=1}^{|\mathcal{L}(t)|} d_i^k \exp(-a_k \kappa_{(i)} h_k(\mathbf{l}_i, \kappa_{(i)})) \quad (5)$$

and  $a_k \in \mathbf{a}$ .

### 3.1.1 The weak hypothesis $h_k$

In the case under consideration the weak hypotheses has the form of a *one-level* decision tree. The test at the root of the tree is a simple check for the presence or absence of a bigram in a given set  $\mathbf{l}_i$ . Let  $b_{(j)}^i$  denote a bigram drawn randomly from  $\mathcal{B}(t)$ . Using this bigram, a weak hypothesis  $h_k$  will be introduced which makes a prediction of the following form [17, 18]:

$$h_k(\mathbf{l}_i, \kappa_{(i)}) = \begin{cases} c_{0r} & \text{if } b_{(j)}^i \notin \mathbf{l}_i \text{ and } \kappa_{(i)} = \kappa_r \\ c_{1r} & \text{if } b_{(j)}^i \in \mathbf{l}_i \text{ and } \kappa_{(i)} = \kappa_r \end{cases} \quad (6)$$

when the link  $\mathbf{l}_i$  is labeled as relevant and

$$h_k(\mathbf{l}_i, \kappa_{(i)}) = \begin{cases} c_{0nr} & \text{if } b_{(j)}^i \notin \mathbf{l}_i \text{ and } \kappa_{(i)} = \kappa_{nr} \\ c_{1nr} & \text{if } b_{(j)}^i \in \mathbf{l}_i \text{ and } \kappa_{(i)} = \kappa_{nr} \end{cases} \quad (7)$$

when it is labeled as non-relevant. In both the above expressions, we assign confidence values in both cases where a bigram is present or not in the set  $\mathbf{l}_i$ . However, intuition rules that we should not assign any confidence value when a bigram is not present ( $c_{0r} = c_{0nr} = 0$ ) [18].

The weak hypotheses  $h_k$  searches all the possible bigrams in  $\mathcal{B}(t)$ . Given a bigram  $b_{(j)}^i$  let the set  $\mathcal{L}(t)$  be partitioned into two disjoint subsets  $\mathcal{L}_m(t)$  where  $m \in \{0, 1\}$ , that is, the sets  $\mathcal{L}_0(t) = \{\mathbf{l}_n | \mathbf{l}_n \in \mathcal{L}(t), b_{(j)}^i \notin \mathbf{l}_n\}$  and  $\mathcal{L}_1(t) = \{\mathbf{l}_n | \mathbf{l}_n \in \mathcal{L}(t), b_{(j)}^i \in \mathbf{l}_n\}$ . For each bigram  $b_{(j)}^i$  and with the minimization of  $\varepsilon(\mathbf{D}_k)$  in mind, the following value is calculated for each label

$$W_b^{mq}(t) = \sum_{i: \mathbf{l}_i \in \mathcal{L}_m(t)} d_i^{(t)} \left[ \text{sign}(\kappa_{(i)}^q) = \text{sign}(b) \right] \quad (8)$$

where  $q$  runs for each available label,  $b \in \{\pm 1\}$ ,  $\llbracket \pi \rrbracket$  equals one if  $\pi$  holds otherwise it is equal to zero and finally,

$$\kappa_{(i)}^q = \begin{cases} +1, & \text{if } q\text{th label was assigned to } i\text{th document} \\ -1, & \text{otherwise} \end{cases} \quad (9)$$

Subsequently, the local extrema minimum values of  $Z_{k+1}$  are achieved when:

$$c_{1q} \stackrel{(\forall q)}{=} \frac{1}{2} \ln \left( \frac{W_{+1}^{1q}(t)}{W_{-1}^{1q}(t)} \right) \quad (10)$$

and since the second derivative is positive, meaning that the minimum value for  $Z_{k+1}$  is achieved when the  $c_{1q}$  are given by Eq. (10).

### 3.1.2 Prediction based on adaboost

After the training phase and when all the weak learners  $h_k$  have been computed the algorithm can proceed to anticipate future user requests and pre-fetch some documents. In doing so let us suppose that currently there is page displayed in the user's browser. For each of the links in that page we evaluate the probable label using Eq. (2). For the links that the sign is positive we estimate the confidence of the prediction using Eq. (3). After this point the pre-fetching mechanism can proceed and pre-fetch some of the linked documents into the local cache.

### 3.2 Test-bed algorithm

The test-bed algorithm relies on the relative frequency of occurrence for the bigrams forming the outbound links. In doing so the anchored text around the visited and the non-visited links are taken into consideration. For that purpose, two counters of the frequencies of appearance are kept and constantly maintained for the bigrams in both cases. The first counter which will be denoted by  $v(b_i)$  records the frequency of appearance of the  $i$ th bigram in the visited links while the second counter is denoted by  $m(b_i)$  and corresponds to non-visited case. Subsequently, the importance of the  $i$ th bigram is given by:

$$w(i) = \frac{v(b_i)}{p} - \frac{m(b_i)}{q}, \quad (11)$$

where  $p$  is the total number of visited links and  $q$  the number of non-visited links. The concept behind Eq. (11) is that a bigram that frequently appears in links that were clicked indicates some importance to the user whereas if it appears in non-visited links signifies repulsion. For example, given two bigrams  $b_{(1)}$  and  $b_{(2)}$  and the following statistics  $v(b_{(1)}) = 30$ ,  $v(b_{(2)}) = 16$ ,  $m(b_{(1)}) = 11$ ,  $m(b_{(2)}) = 8$ ,  $p = 105$  and  $q = 55$  then the relative importance for  $b_{(1)}$  is  $w(b_{(1)}) = 0.08571$  whereas for  $b_{(2)}$  it is  $w(b_{(2)}) = 0.00692$ . That makes the first term far more important according to the user's past preferences and this should be reflected by the weighting mechanism that is applied to new links.

So, the weight of the link according to the contribution of the bigrams around it is:

$$T^{new} = \sum_{i=1}^{|new|} w(i). \quad (12)$$

### 3.3 General remarks

Some final notes concerning the pre-fetching algorithms in general must be stated here before proceeding to the next subsection:

- If there is a fast transition between web pages then the viewed pages are not taken into consideration. This is

needed in order to ensure that the user has spend some time either reading or by simply searching around for something interesting in that particular web page.

- The number of links to be pre-fetched is bounded by the following threshold:

$$\lfloor \frac{\beta \mu(t)}{\mu(s)} \rfloor, \quad (13)$$

where  $\beta$  corresponds to the total bandwidth available,  $\mu(t)$  is the median operator over the time spend by the user in each page,  $\mu(s)$  is the median operator over the size of the visited web pages and finally  $\lfloor \cdot \rfloor$  denotes the floor function.

- Pre-fetching occurs when the weighting mechanism has assigned weights to each of the outbound links and it takes place during the browser's idle time. During that time multiple threads are employed and each thread receives one link to pre-fetch. After a thread has completed the pre-fetch of the URL that was assigned to it, a new URL is assigned to it and so on.
- If a user-driven event occurs before the completion of the pre-fetch session then all the threads are told to stop executing immediately in order to allow the browser to utilize the entire available bandwidth. For bandwidth management and system resource utilization the interested user can also consult [19].

## 4 Assessment of the pre-fetching capabilities

The comparison of the capabilities for pre-fetching algorithms, particularly the content-based ones, poses up to a degree some difficulties. All the evaluation techniques rely either on artificial data sets or access logs. In the former case, the content based algorithms are useless due to the lack of textual content in the generated sets. In the later case, a problem that often emerges is that due to the dynamic nature of the Internet the content of the web pages tend to change continuously. Therefore, access logs that have been used in the past may not be prime candidates for present studies either due the change of the content or even worse due to the total extinction of a portion of the logged web pages. Due to the above reasons a trace of the web activity for 10 volunteers has been gathered. The trace is divided into two sets, a *training set* which was used to build the personalized profiles for the users and a *test set* which was used in evaluating the performance of the proposed algorithm.

### 4.1 Training set

The training trace consists of nearly 102000 individual web pages. After the processing of the trace, an individual user profile was created for each of the volunteers and for each of the proposed algorithms based solely on the web pages

that comprised his or her activities. The profiles created in the *training phase* of the algorithm were exploited afterwards in the *testing phase*. In the testing phase each volunteer was asked to use a custom-made GUI that resembles a web browser in order to surf the Internet. They were instructed to follow their normal every-day activities regarding their surfing habits. The profiles of each particular volunteer was used by the stand-alone application in order to pre-fetch a portion of the outbound links for each of the web pages that this particular user visited.

## 4.2 Testing set

During the testing phase an access logs consisting of 12500 individual web pages was captured. This trace is employed in a performance evaluation that relies on the *cache-hit ratio* and the *fractional latency reduction* that the algorithms achieve compared to the *fractional network traffic* that might be caused to the network [20].

*Cache-hit ratio* is the ratio of pre-fetched pages that a user requested to all the pages the pre-fetch agent retrieved and represents the accuracy of the prediction.

*Fractional latency reduction* is the ratio between the decrease due to pre-fetching of the observed UPL without a caching system from the UPL with a caching system to the observed UPL without caching.

*Fractional network traffic* is the ratio between the amount of bytes transmitted from a web server to the user's client to the total number of bytes requested.

Figure 1 depicts the cache-hit ratio curves for each of the pre-fetching algorithms. The in question figure depicts the average cache-hit curves for all the volunteers in predefined steps of the outbound links volume. From the figure it is evident that all curves will converge to the same cache-hit point when all the outbound links will be pre-fetched. This is inevitable since the number of useful outbound links is not a function of the pre-fetching mechanism employed, rather than the personal preferences of each user. Furthermore, the adaboost algorithm exhibits better performance than its prime candidate in each pre-fetch volume, with a maximum difference of 7.1% higher cache-hit rate which is achieved at a 30% pre-fetch volume.

Finally, as it can be seen in Fig. 2(a), prefetching does reduce network latency in all pre-fetch volumes whereas the bandwidth overhead is more or less the same for all the algorithms (Fig. 2(b)). The latency is reduced from around 28% when only 10% of the links are retrieved to nearly 70% when 90% of the links have been pre-fetched. At the same time the prime candidate achieves similar but less satisfactory results with a difference in the reduction of the latency in the range from 28% to 67%.

## 5 Conclusions

In this paper we provided two novel transparent and speculative algorithms to model the user's behavior when surf-

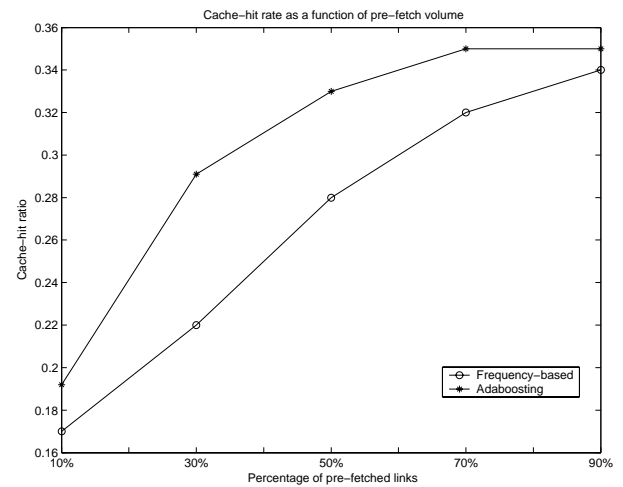


Figure 1. The average cache-hit ratio curves for each of the four pre-fetching algorithms in predefined step of the volume of pre-fetched outbound links.

ing the Internet. The proposed modeling algorithms are based on the frequency of occurrence for selected bigrams forming the visited web pages and a variant of the adaboost algorithm. The user behaviour model is used for the prediction of future actions. For the prediction of the next link to be visited the algorithm uses the anchored text around the outbound links and assigns weights to each of these links. Following, the algorithms pre-fetch some of the linked documents and stores them locally in a cache in an effort to reduce the UPL. The proposed algorithms has been tested against two other algorithms and demonstrated superior performance in predicting the user's future requests in the cache-hit ratio while slightly increasing the bandwidth overhead.

## References

- [1] Zona Research, "The economic impacts of unacceptable web-site download speeds," April 1999.
- [2] C. Aggarwal, J. Wolf, and P. S. Yu, "Caching on the world wide web," *IEEE Trans. Knowledge and Data Eng.*, vol. 11, no. 1, pp. 95–107, 1999.
- [3] J. Shim, P. Scheuermann, and R. Vingralek, "Proxy cache algorithms: Design, implementation, and performance," *IEEE Trans. Knowledge and Data Eng.*, vol. 11, no. 4, pp. 549–562, 1999.
- [4] J. Wang, "A survey of web caching schemes for the internet," *ACM Computer Communication Review*, vol. 29, no. 5, pp. 36–46, October 1999.
- [5] B. Berendt and M. Spiliopoulou, "Analysis of navigation behavior in web sites integrating multiple information systems," *The VLDB Journal*, vol. 9, no. 1, pp. 56–75, 2000.

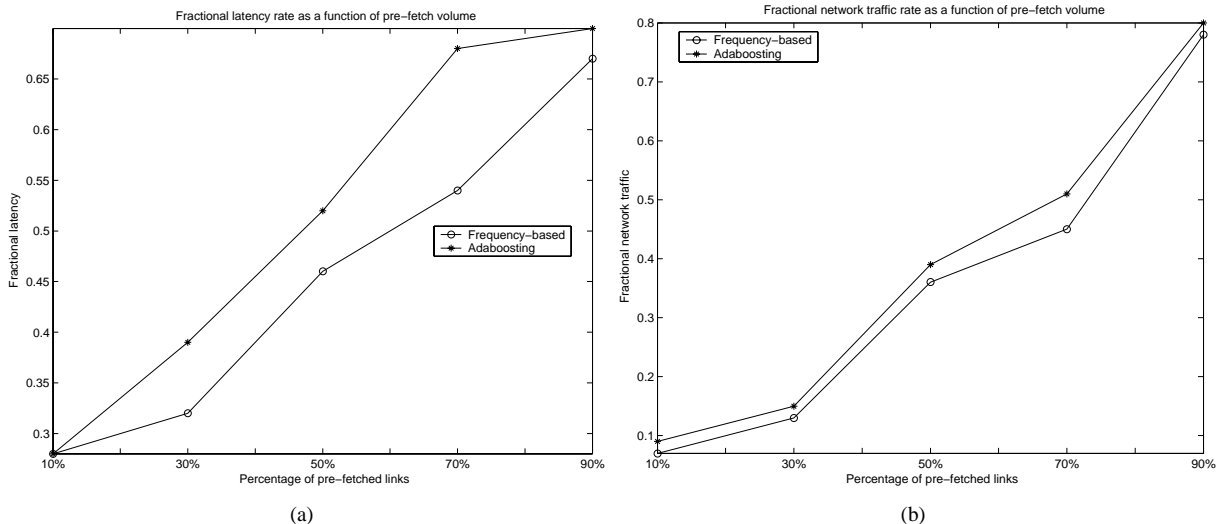


Figure 2. (a) The fractional latency reduction ratio for given volumes of pre-fetched outbound links. (b) The fractional network traffic ratio for specified volumes of pre-fetched outbound links.

- [6] Q. Yang and H. H. Zhang, "Web-log mining for predictive web caching," *IEEE Trans. on Knowledge and Data Eng.*, vol. 15, no. 4, pp. 1050–1053, 2003.
- [7] L. Fan, P. Cao, and Q. Jacobson, "Web prefetching between low-bandwidth clients and proxies: Potential and performance," in *Proc. of the Joint Int. Conf. on Measurement and Modeling of Computer Systems (SIGMETRICS99)*, 1999, pp. 178–187.
- [8] M. Géry and H. Haddad, "Evaluation of web usage mining approaches for users next request prediction," in *Proc. of the 5th Int. Workshop on Web Information and Data Management (WIDM 2005)*, 2003, pp. 74–81.
- [9] V. N. Padmanabhan and J. C. Mogul, "Using predictive prefetching to improve world wide web latency," *Computer Communication Review*, vol. 26, no. 3, pp. 22–36, 1996.
- [10] J. I. Khan and Q. Tao, "Partial prefetch for faster surfing in composite hypermedia," in *Proc. of the 3rd USENIX Symposium on Internet Technologies USITS01*, 2001, pp. 13–24.
- [11] R. R. Sarukkai, "Link prediction and path analysis using markov chains," in *Proc. of the 19th International World Wide Web Conf.*, 2000.
- [12] P. Pirolli and J. E. Pitkow, "Distributions of surfers' paths through the World Wide Web: Empirical characterization," *World Wide Web*, vol. 2, no. 1–2, pp. 29–45, 1999.
- [13] M.F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.
- [14] D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*, Cambridge, MA: MIT Press, 1999.
- [15] F. Sebastiani, "Machine learning in automated text categorization," *ACM Computing Surveys*, vol. 34, no. 1, pp. 1–47, 2002.
- [16] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in *Proc. of the 14th Int. Conf. on Machine Learning, ICML-97*, 1997, pp. 412–420.
- [17] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, vol. 37, no. 3, pp. 297–336, 1999.
- [18] R. E. Schapire and Y. Singer, "BoosTexter: A boosting-based system for text categorization," *Machine Learning*, vol. 2/3, no. 39, pp. 135–168, 2000.
- [19] W.-S. Li, K. S. Candan, and D. Agrawal, "Method and apparatus for intelligent resource utilization for web content fetch and refresh," 2004, US Patent No. 6,701,316.
- [20] B. D. Davison, *The design and evaluation of web prefetching and caching techniques*, Ph.D. thesis, State University of New Jersey, 2002.