

User behaviour modeling and content based speculative web page prefetching

Abstract

This paper provides a transparent and speculative algorithm for content based web page prefetching. The algorithm relies on a profile based on the Internet browsing habits of the user. It aims at reducing the perceived latency when the user requests a document by clicking on a hyperlink. The proposed user profile relies on the frequency of occurrence for selected elements forming the web pages visited by the user. These frequencies are employed in a mechanism for the prediction of the user's future actions. For the anticipation of an adjacent action, the anchored text around each of the outbound links is used and weights are assigned to these links. Some of the linked documents are then prefetched and stored in a local cache according to the assigned weights. The proposed algorithm was tested against three different prefetching algorithms and yield improved cache-hit rates given a moderate bandwidth overhead. Furthermore, the precision of accurately inferring the user's preference is evaluated through the recall-precision curves. Statistical evaluation testifies that the achieved recall-precision performance improvement is significant.

Keywords: Link prefetching, user behavior modeling, bigrams.

I. INTRODUCTION

The growing popularity of the *World Wide Web* (WWW) in accordance with an increase in the size of the web pages yields long *User Perceived Latency* (UPL) for documents hosted in some of the web servers. Over the time it has become evident that short UPL has positive effects on both the users and the owners of the web sites since an increase of the satisfaction perceived by a user is reflected also to the site's owner. Moreover this satisfaction is crucial for the survivability and prosperity of any on line merchant. It is proven that the "success" of a web site is correlated to the time it takes to upload their

material to the user's browser. A rather old and outdated but still useful study conducted in 1999 by Zona Research Inc. provides evidence that if a Web site takes more than eight seconds ($\sigma = 2$ sec) to download then a 30% of the visitors is more likely to leave the site [Res99].

Many solutions have been proposed and implemented in order to decrease the UPL. The solutions range from the availability of faster Internet connections for the users and alternative communication technologies to faster web servers and increased ISP bandwidth either from a single backbone provider or multiple providers. Furthermore, the usage of proxy servers by the ISPs for caching of "popular" documents also proved to lessen the UPL. Finally, server side proxies and distributed servers for traffic balancing are also used and can substantially improve typical response times. Still, the UPL exist and is amplified when the user is visiting web pages and documents that never visited in the past.

The UPL can be reduced from the user's side as well without extra costs through the usage of the browser's cache [AWY99], [SSV99]. In this solution the documents that are more likely to be accessed are *prefetched* to the browser's cache. When a request takes place for one of these documents a *cache-hit* occurs thus avoiding some portion of the possible delays. Given the previous, a crucial issue is to effectively predict the following requests and subsequently to model the user's actions over time. If the next request can be determined accurately then the UPL tends to zero. Furthermore, there is no bandwidth misuse. Unfortunately, since the probability of an accurate prediction is almost always below one, more than one pages must be prefetched in order to achieve low UPL. In that case some prefetches may and will never be used which will result in bandwidth overhead and increased load on the web servers.

The algorithm proposed in this paper is based on the assumption that the textual information for both the visited pages and the followed links is highly relevant to the preferences of the user. That is, when the user selects a hyperlink, that decision relies on the textual description provided around the link. For example, Fig. 1 is an excerpt from the Open Directory project. A user interested in the *Fermat primes* will select the link with the textual description *Generalized Fermat prime search* and probably skip any other link.

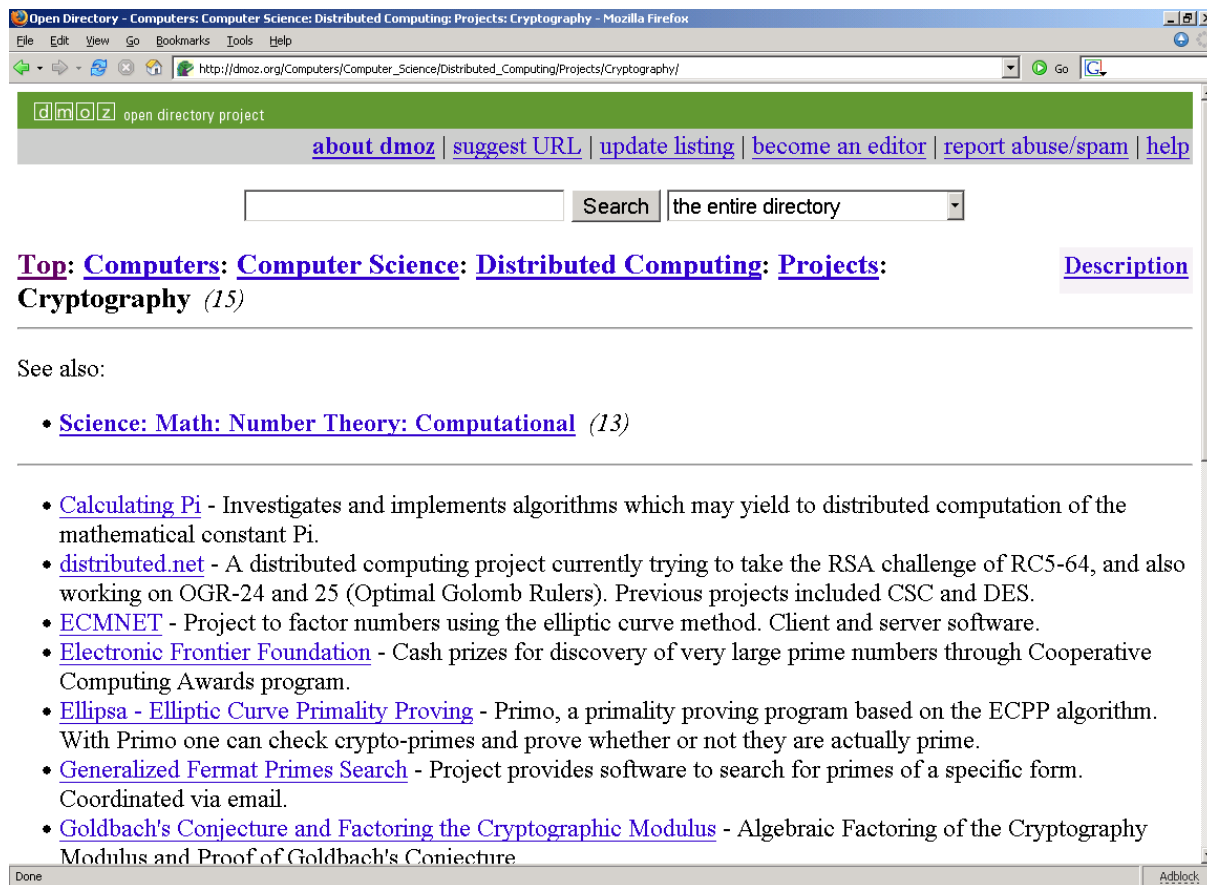


Fig. 1. A sample web page from the *Open Directory* project.

Therefore, when a user visits a web page and spends a considerable amount of time either reading or looking around for valuable information then the textual content of the page is the prime candidate for the user's *Region Of Interest* (ROI). Furthermore, the followed links also contain text relevant to the user's ROI. Therefore, it is justified to assume that the ROI comprises of the text forming the visited pages and the text anchored around the followed links. Hence, when modelling the user's behaviour one can rely on the frequency of occurrence of some carefully selected terms.

Moreover, studies conducted both for *machine translation* and *Information Retrieval* (IR) tasks have shown that sequences of terms yield superior performance compared to individual terms [Seb02], [MS99] in capturing the latent content of a document. So, instead of building the profile on individual terms one should create a profile using sequences of terms. In the previous example the user was interested in the terms *Fermat* and *primes*. Instead of using these two terms, one can replace them by the sequence <

Fermat primes>. Such a sequence is usually addressed as a *bigram*. The notion n -gram can also be found in the literature when a sequence of n terms are regarded as a single object. Subsection III-B will provide a short description of the n -gram modelling in general and it will also supply an explanation as to why the bigram model was chosen to model the user's ROI instead of a different, higher order, n -gram model ($n = 3, 4, \dots$).

After the creation of the profile the prefetching mechanism can determine which outbound link is more probable to be requested in the near future. 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 frequencies in the profile are used and individual weights are computed for these links. Subsequently, the algorithm retrieves a portion of the linked documents according to the assigned weights. In case of equiponderant links the proposed algorithm uses two variations of the *PageRank* algorithm [BP98] to solve the ambiguities.

It is crucial here to explain the difference between the terms *linked text* and *anchored text*. In the following example:

“If you are interested in Fermat primes click *here*.”

the html code associated would look something like this:

```
If you are interested in Fermat primes  
click <a href="http://aaa.bbb.ccc">here</a>
```

In that case the linked text would correspond to the word *here* which clearly is not very informative and the anchored text would have been the entire sentence. The previous example, although is a bit far fetched, clearly depicts the draw backs associated with using only the linked text. The anchored text is more informative and is an important factor in the proposed method. The only limitation is the size of the sentence surrounding the link. Subsections III-A and III-B will explain the filtering processes undertaken on the anchored text in order to reduce the dimensionality of the modeling space.

In what follows, section II provides a brief description of the prefetching literature and algorithms devised and implemented so far. Section III covers the proposed prefetching algorithm along with the

creation and the updating of the user's profile. Section IV describes the two versions of the PageRank algorithm employed in disambiguation and finally, section V provides the experimental results for the evaluation of the proposed algorithm.

II. RELATED WORK

The UPL can be reduced with the exploitation of one or more cache repositories located either remotely in a proxy server or the web server hosting the documents under consideration or locally by the browser's cache or a local proxy server. Local proxy servers are helpful in cases where two or more users have access through the same connection. The usage of a cache can reduce the overall traffic and lessen the need for the client to contact a web server when the documents are stored in the cache. The fact that a proxy server stores documents without taking any initiative for predicting which documents are more probable to be requested (*passive mode*) is a major drawback. This drawback can be lifted with the usage of algorithms that predict the future demand of particular documents (*active mode*). Subsequently, the documents that are more probable to be requested in the future are prime candidates for caching.

Figure 2 depicts the three alternative methods of client-server communication. In Fig. 2(a) the client requests a document from the cache and the cache either returns the requested document to the client or forwards the request to the Web server. In Fig. 2(b) the client requests the desired document from the cache (solid line) whereas the prefetching algorithm requests a plethora of relative documents (discontinuous line) that have high probability to be requested in the near future. Finally, in Fig. 2(c) the web server returns to the cache repository the requested document and the preloading algorithm, which works in parallel with the server, "pushes" to the intermediate cache some extra documents with the anticipation that they might be requested in the future.

Very useful overviews on caching and prefetching techniques can be found in [Wan99]. Combinations of caching and prefetching have been studied in [WC96], [FS00], [YZL01]. Lately, prefetching has gained much attention due to some inherent limitations in caching. Markatos *et al.* proposed the Top-10 prefetching algorithm which is a hint based method [MC98]. Other hint based prefetching techniques can

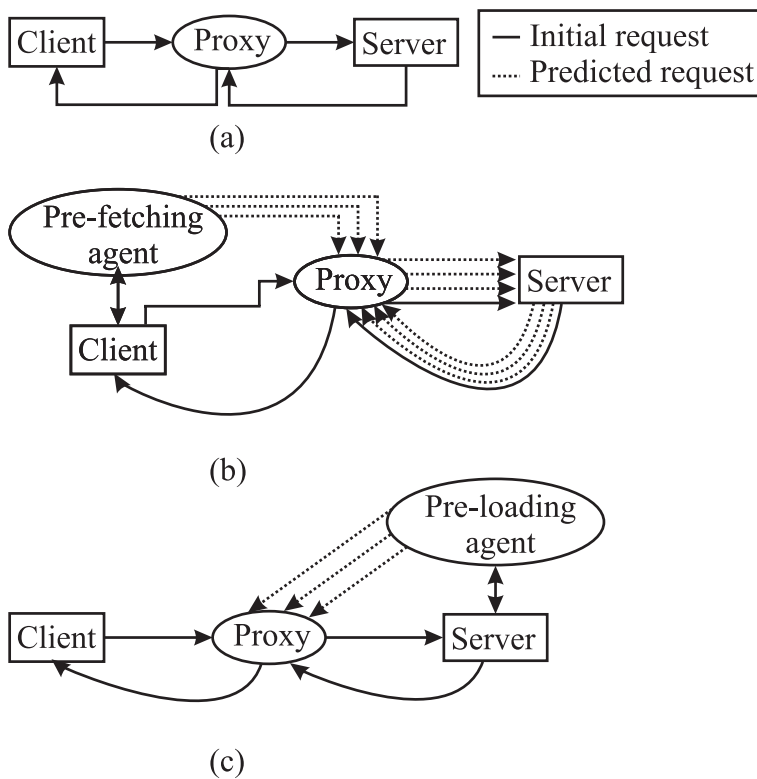


Fig. 2. The three different methods of transaction between a client and a web server: (a) through cache repository, (b) a cache repository and a prefetching agent working in parallel with the client, (c) a cache repository and a preloading agent working in cooperation with the web server.

be found in [CKR98], [PP99b]. Bestavros in [Bes95] proposed the utilization of a document transition matrix. Data mining techniques have also been used to extract patterns from Web logs [BS00], [YZ03], [PHMAZ00]. Moreover, the *Prediction by Partial Match* has also been employed extensively in [FCJ99], [PM99]. Furthermore, recommendation systems and web usage mining techniques can be used in order to provide hints to a prefetching agent over to which links are more probable to be requested [MDLN01], [GH03], [GST02]. In [PM96] 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 [KT01]. Discrete Markovian techniques have been applied on the history of Web page references to recognize patterns in the activity of the use [Duc99], [Sar00]. Pirolli and Pitkow in [PP99a] and [PP99b] propose the usage of high order Markov Models and n -gram modeling in order to predict the behaviour and future actions of a user. Furthermore, they showed that

longer sequence of user visits can lead to a reduced model in size without affecting the ability of accurate predictions. In [DK01] a combination of different order Markov models is used to devise a model with lower state-space complexity. An extension to the discrete model is the continuous Markov chain found in [KW98].

The above methods rely on the web server in order to determine the importance of a web page. In the client based approaches, on the other hand, the most primitive method is to prefetch the embedded links in a top-to-bottom order regardless of the likelihood to be requested [CY97]. A similar approach is to randomly prefetch links. Others prefetch links from the currently requested page [Inc02], [Cor02a], [Cor02b], [Kle99]. Another client-side alternative is to do everything but prefetch that is, to resolve the DNS in advance, connect in advance to the Web server, and even warm up the Web server with a dummy request [CK00]. Chan in [Cha99] proposes a non-invasive approach to construct user profiles, that incorporates content, linkage, as well as other factors. Finally, Yang *et al.* in [YSG02] considered various approaches to hypertext classification.

It must be noted here that the benefits of any of the above 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 prefetched 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.

III. PROPOSED ALGORITHM

This paper provides a novel *transparent* and *speculative* prefetching algorithm for Web documents on behalf of the user. The algorithm is transparent since the user is not involved in the selection of the documents to be prefetched and it is speculative since the future requests are predicted based on knowledge acquainted during past experience. For that reason a profile is created based on the user's *prior behavior*. Subsequently, the information contained in the profile is utilized in deciding which of the outbound links

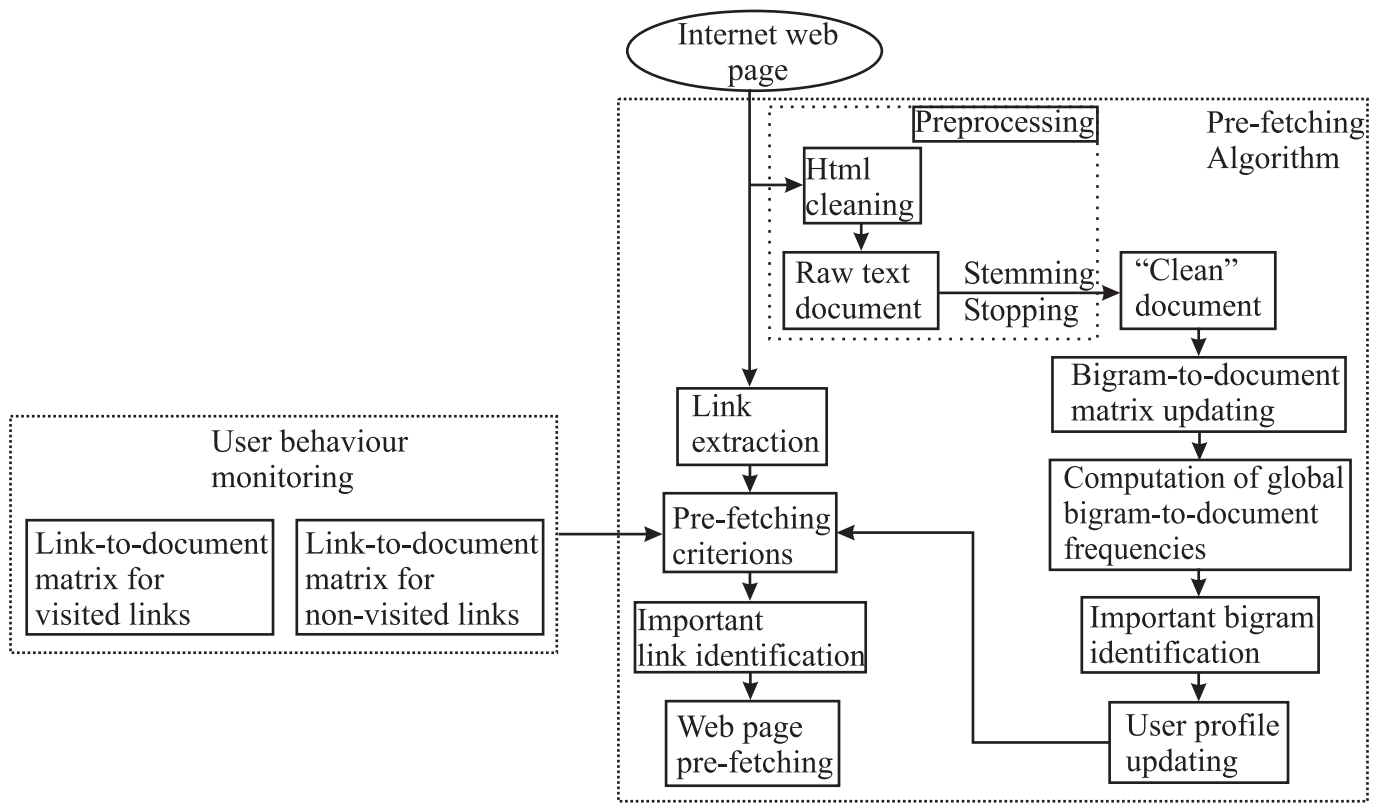


Fig. 3. Block diagram of the prefetching algorithm.

to be retrieved for probable latter request. The entire process described in the present section can be seen in Figure 3.

A. Preprocessing

The first step towards the formation of the profile is the retrieval of the starting page for the user. A *Perl* script based on the *libwww* library is used for that purpose. Subsequently the retrieved document is preprocessed through the following stages:

- Identification of the outbound links and the anchored text around them.
- Elimination of the links that the browser definitely should not prefetch. That category comprises of pages with time critical data and pages that are already in the cache either from an earlier access or prefetch.
- Removal of the HTML tags and entities. The information regarding the outbound links is kept in a separate file for later usage.

- 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.
- Text cleaning is applied by removing some common English words such as articles, determiners, prepositions, pronouns, conjunctions, complementizers, abbreviations. Non-English frequent terms were also removed by applying stopping.
- Subsequently stemming is performed. Stemming refers to the elimination of the word suffixes so that the vocabulary shrinks, although it keeps the informative context of the text. The commonly used Porter stemmer was applied [Por80].

It should be noted here that the proposed algorithm can easily handle any type of web pages regardless of the *charset* parameter. That is, depending on the *charset* parameter different stemming algorithms can be utilized. Moreover, the usage of stemming can be totally ignored although a degradation of the prefetching effectiveness is then unavoidable.

After preprocessing the resulted web page contains probably a plethora of stems. Some of these stems may actually characterize the user's interests. In order to identify which of these terms should be associated with the user's profile we will use the so-called *low-to-medium* "law" from the IR community. According to this law the terms whose document frequency is low-to-medium are the most informative ones [MS99], [Seb02]. The above implies that the stems with high frequency are not informative regarding the actual content of the documents; they rather help in the formation of sentences and therefore can be eliminated. It can be seen in the literature that the above term selection technique can reduce the dimensionality of the modeling space by a factor of 10 or even more without any loss in discriminative power of the resulting textual information [YP97].

B. Contextual statistics evaluation

In order to infer future events, we first need to find other features that predict it. Here, we assume that past behaviour is a good guide to what will happen in the future. In doing this, we effectively partition the available data into equivalence classes. In other words, dividing our data into distinct *bins* yields

improved discriminative power. The task of predicting one word or stem from its previous words (*history*) is a stochastic problem. And since we cannot consider every history separately we need a method to group similar histories. One possible grouping way is by making a *Markov assumption* that all histories with the same $(n - 1)$ words are placed in the same equivalence class. This yields an n -gram model.

The cases of models that are usually used are for $n = 2, 3, 4$, and these models are referred to as *bigram*, *trigram* and *four-gram*. The usage of n -grams has prevailed over the unigrams (individual terms) since it has been proven in a number of experiments that the n -grams can capture collocations. Example of collocations are: *strong tee*, *make up*, *weapons of mass destruction* etc. Collocations are important because they provide latent information for the contextual meaning of any given text.

The size of the history (model) is controlled by the size of the textual data available. Rosenfeld in [Ros00] reports that:

The value of n trades off the stability of the estimate (*i.e.* its variance) against its appropriateness (*i.e.* bias). A trigram ($n = 3$) is a common choice with large training corpora (millions of words), whereas a bigram ($n = 2$) is often used for smaller ones.

We should note here though that for the trigram model the above figure (millions of words) is underestimated. The currently accepted figure is in the range of 10^7 to 10^8 of words. In this paper, since the collection of training data were not expected to exceed the above trigram threshold 10^7 the *bigram* model was adopted. In this case only one preceding stem is used to encode each stem [MS99]:

$$P(o_i|o_{i-1}) = \frac{\eta(o_{i-1}, o_i)}{\eta(o_{i-1})}, \quad (1)$$

with $\eta(o_{i-1}, o_i)$ and $\eta(o_{i-1})$ denoting the number of observed bigrams (o_{i-1}, o_i) and stem types o_{i-1} , respectively, in the collection of processed documents. In the following, the discussion will be based on the conditional probabilities computed for the generic observed bigram $b_l = (o_{l-1}, o_l)$.

C. Profile creation

For the creation of the profile the algorithm uses a variation of the low-to-medium law mentioned earlier. Let $A_{N,M}(t)$ denote the bigram-to-document indicator matrix, that is, $A_{N,M}(t) = (a_{ijt})$ where N denotes the number of bigrams and M denotes the number of documents visited by the user until the time instant t . Furthermore, a_{ijt} is a boolean valued variable where:

$$a_{ijt} = \begin{cases} 1 & \text{if the } i\text{th bigram is present in the } j\text{th document} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Let also $\mathcal{F} = (f_{b2d}(b_1, t), f_{b2d}(b_2, t), \dots, f_{b2d}(b_N, t))$ denote the global bigram-to-document frequencies, where $f_{b2d}(b_i, t) = \sum_{k=1}^M a_{ikt}$, that is the number of documents in which the i th bigram is present at least once.

Obviously, bigrams that are found in almost every page that the user visits should correspond, with some uncertainty, to the user's preference. For example, if the user is interested in the *Fermat primes* and this bigram is found in every page ever visited then it should be part of the user's profile. If, on the other hand, the above bigram was found just once in all the visited pages then it does not sound to be of a particular interest to the user. Therefore, it can easily be removed from the profile.

In selecting the most important bigrams, the frequencies in the vector \mathcal{F} are arranged into descending order which produces a graph similar to Fig. 4. By rejecting the low frequent bigrams (grayed area) we form the user's profile. Other term selection methods that can be applied as well are: the DIA association factor [FB91], chi-square [SSV00], [YL99], information gain [Mla98], mutual information [RS99].

A remark that should be made at this point is that both the thresholding step that was performed at the end of subsection III-A along with the rejection of the non-frequent bigrams that was mentioned in the present subsection are performed in order to reduce the dimensionality of the modelling space and thus speed up the entire process.

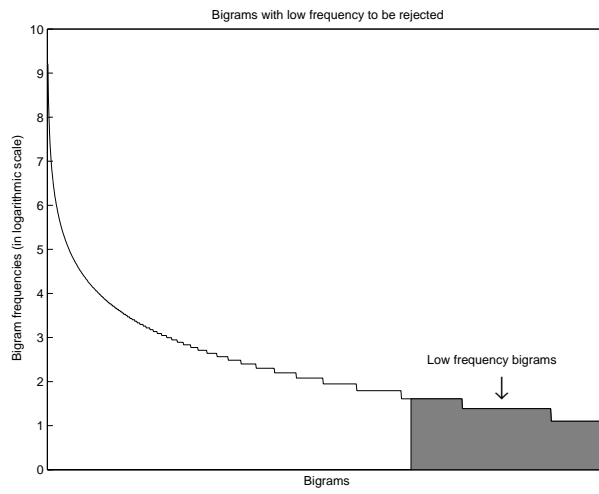


Fig. 4. The bigrams consisting the web pages viewed by the user arranged into descending frequency order. The bigrams with low frequency (grayed area) will be rejected from the rest of the process.

D. Prediction algorithm

Let $\mathcal{B}(n) = \{b_1, b_2, \dots, b_{|\mathcal{B}(n)|}\}$ denote the set of bigrams comprising the user's profile. Let also $\mathcal{X}_s = \{b_1^s, b_2^s, \dots, b_{n_s}^s\}$ denote the bigrams associated with the s th outbound link where $n_s = |\mathcal{X}_s|$. In order to assign a weight to the s th link we will sum the frequencies of the bigrams comprising the particular link:

$$w_1^s = \frac{\sum_{i=1}^{n_s} f_{b2d}(b_i^s, t)}{n_s}. \quad (3)$$

Furthermore, the trends in the user's interest are reflected by the path of visited links. In order to grasp that behavior the anchored text around the visited and the non-visited links must be taken into consideration. In doing so the algorithm keeps two counters of the frequencies of appearance for the bigrams in both cases. The first counter which will be denoted by $v_{b2l}(b_i, t)$ records the frequency of appearance of the i th bigram in the visited links while the second counter is denoted by $m_{b2l}(b_i, t)$ and corresponds to non-visited case. Subsequently, the importance of the i th bigram is given by:

$$\frac{v_{b2l}(b_i, t)}{p} - \frac{m_{b2l}(b_i, t)}{q}, \quad (4)$$

where p is the total number of visited links and q the number of non-visited links. So, the weight of the

link according to the contribution of the bigrams around it is:

$$w_2^s = \sum_{i=1}^{n_s} \left[\frac{v_{b2l}(b_i^s, t)}{p} - \frac{m_{b2l}(b_i^s, t)}{q} \right]. \quad (5)$$

The reasoning behind the counters $v_{b2l}(b_i, t)$ and $m_{b2l}(b_i, t)$ is similar to the reasoning provided in subsection III-C. The more often a bigram is found in a visited link the more important to the user seems to be. On the other hand, the more often a bigram is found in a non-visited link the less important to the end user seems to be. Therefore, by using Eq. (4) we can estimate the actual importance to the user for any given bigram.

Finally, the total weight of the s th link is:

$$T_s^1 = w_1^s + a(t) w_2^s \quad (6)$$

where $a(t)$ is a monotonically decreasing scalar-valued “*learning-rate factor*” which asymptotically tends towards unity. The parameter $a(t)$ is useful during the first iterations of each training epoch since the bigram frequencies in the vector \mathcal{F} outweighs the frequencies in both the counters $v_{b2l}(b_i, t)$ and $m_{b2l}(b_i, t)$. This is attributed to the fact that the population of the bigrams forming the documents is always larger to the bigram population in the anchored text around the links. So, if its for the counters $v_{b2l}(b_i, t)$ and $m_{b2l}(b_i, t)$ to participate in the estimation of the total weight of any given link we need to balance between w_1^s and w_2^s and parameter $a(t)$ plays that role.

E. User input

Unfortunately the extraction of a reliable user profile is extremely difficult due to the complexity of natural language. For this reason user input can be requested in order to elevate the overall performance. More specifically, the user can suggest a list of keywords which can be enriched through the usage of a thesaurus. The keywords should trigger the algorithm to grand extra weight to the links containing them. Let $k_i, i = 1, 2, \dots, N_k$ denote the user defined keywords, where N_k is the number of keywords. For each of the keywords the algorithm keeps also two counters. The first one, $\eta_v(k_i)$, denotes the frequency of

the i th keyword in the visited links whereas the second counter, $\eta_{nv}(k_i)$, denotes the frequency in the non visited. Subsequently, the weight for the i th keyword is:

$$\frac{\eta_v(k_i)}{p} - \frac{\eta_{nv}(k_i)}{q}. \quad (7)$$

But since the user keywords are more important than the automatically extracted bigrams the following scaling parameter is used to empower their importance:

$$\beta(t) = \max \left(f_{b2d}(b_i^s, t), \left[\frac{v_{b2l}(b_i^s, t)}{p} - \frac{m_{b2l}(b_i^s, t)}{q} \right] \right). \quad (8)$$

So, the total weight for the s th link using the user's input is:

$$T_s^2 = T_s^1 + \frac{\sum_{i=1}^{N_k} \beta(t) \left[\frac{\eta_v(k_i)}{p} - \frac{\eta_{nv}(k_i)}{q} \right] I(k_i, X_s)}{\sum_{i=1}^{N_k} I(k_i, X_s)}, \quad (9)$$

where $I(k_i, X_s)$ is an indicator factor:

$$I(k_i, X_s) = \begin{cases} 1, & \text{if } k_i \in X_s \\ 0, & \text{if } k_i \notin X_s \end{cases}. \quad (10)$$

F. Important comments

With the completion of the sections dealing with the weight estimation for each link we should provide some final notes concerning the prefetching algorithm in general:

- If there is a fast transition between web pages then the viewed pages are not taken into consideration. This is in order to ensure that the profile comprises of bigrams found in pages that the user has spend some time either reading or by simply searching around for something that might be interesting.
- The concept behind Eq. (4) 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. The same mechanism is employed when weighting the keywords in order to ensure that the provided keywords are of the user's actual interest and that they were not supplied mistakenly by the user.

- The user’s input is not important to the algorithm which can decide on the importance of the links based on the value T_s^1 in Eq. (6). But, if some input is provided then the value T_s^2 in Eq. (9) can be used instead.
- The number of links to be prefetched is bounded by the following threshold:

$$\lfloor \frac{\beta \mu(t)}{\mu(s)} \rfloor, \quad (11)$$

where β 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.

- Prefetching occurs when the weighting mechanism has assigned weights to each 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 prefetch. After a thread has completed the prefetch 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 prefetch session then all the threads are told to stop executing immediately in order to allow the browser to utilize the entire available bandwidth.

IV. RESOLVING AMBIGUITIES

When a tie occurs in the weights assigned to two or more links then the algorithm cannot “guess” the order in which to assign the links to the threads. And although multiple threads are employed for prefetching it is probable to have less free threads than links with the same weight to be prefetched. In that case a supplementary module is utilized by the algorithm to disambiguate this kind of issues. The module relies on the PageRank algorithm which originated from the Stanford University and is now used by Google [BP98].

PageRank is a numeric value that represents the importance of a page on the web according to the rest of the indexed pages in the Internet and is roughly the output of a voting system between all the web pages. This value is computed in the following way: when one page links to another page, this denotes

TABLE I

FIVE OUTBOUND LINKS WITH EQUAL WEIGHTS.

Candidate link	Local PageRank	Google PageRank	Retrieval sequence
A	20.5	-	1 st
B	15	7	2 nd
C	15	6	3 rd
D	13	-	4 th
E	-	-	5 th

that the page with the inbound link is important to the page with the outbound link due to its content. So, the page with the outbound link votes for the page with the inbound link. The number of votes that are casted to a particular page signify the importance of the particular page. Furthermore, the importance of one page voting for another affects the importance of the casted vote. In this paper, PageRank is used to select the most important links among the ones that the proposed algorithm casted the same weight.

We should note here that we are not interested in the actual PageRank value provided by Google since this value corresponds to the importance of a link according to the portion of the Internet indexed by their spiders. Rather, we aim at the importance of a links according to the user's history. That is, the history forms a miniature of the Internet and the *local PageRank* variant computes a numerical value which denotes the importance of a link according to the rest of the pages in the user's history. This value is used to disambiguated some of the emerged uncertainties. In case that the uncertainties remain due to ties in the *local PageRank* values then the Google's PageRank values are used. In the case of a weight ambiguity with a document that was never seen by the *local PageRank*, this particular document is set to be retrieved after the rest of documents with equal weight has been cached locally.

In Table I we provide an example on resolving ambiguities with five outbound links. The links are labeled A, B, C, D and E respectively and has been given equal weights from the weighting section of the algorithm (see Eq. (9)). The *local PageRank* module scores each of the links according to the voting process among the rest of the web pages in the user's history. The scores can be seen in the second

column of the table. Clearly document A will be the first one to be retrieved while document E will be the last one since it was never seen by the *local PageRank*. For documents B and C, where the ambiguity still exist (*local PageRank* equal to 15) the Google's PageRank is retrieved. After the completion of the above step the order of retrieval for the five documents can be seen in the last column of Table I. With the completion of the last step the algorithm can proceed and prefetch the links.

V. ASSESSMENT OF THE PREFETCHING CAPABILITIES

The comparison between prefetching algorithms, particularly the content-based ones, poses 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, 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 usable in present studies either due the change of the content or even worse the total extinction of a portion of the logged web pages.

Due to the above reasons a custom-made trace for the web activity of 55 volunteers was gathered over a period of seven months in 2005. The trace is divided into two sets, a *training* and a *test* set. The training set was used to build a personalized profile for each individual user whereas in the testing phase the performance of the proposed algorithm is assessed against three test-bed algorithms which are:

- *top-down* algorithm, corresponding from the first link visible in the top of the page to the bottom
- *random selection* of the links in the web page
- *bigrams in link description*

The algorithm addressed as "*bigrams in link description*" differs from the proposed algorithm in the fact that it does not use the *anchored text around the links* rather than the text in the link description.

It must be noted here that each user has a personalized profile build from the web pages visited by him or her during the training phase. This profile is the starting point for the testing phase where new and unseen events are used for evaluation of prefetch. Moreover, in both the training and testing phase the procedures followed were the same, namely: (a) preprocess the starting page, (b) extract links, (c) start

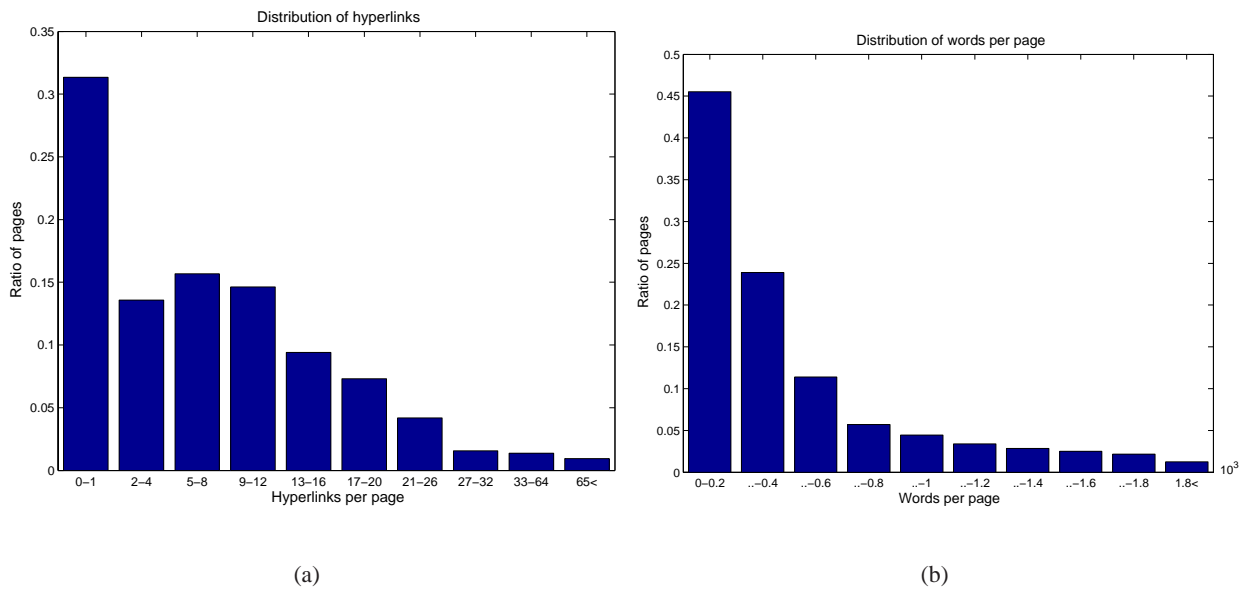


Fig. 5. (a) The distribution of the outbound links. (b) The distribution of the length of the web pages.

prefetch, (d) break prefetching if a user generated event occurs and finally (e) update the profile using the newly visited links and the bigrams forming them.

A. Training set

The training trace consists of nearly 435000 individual web pages. Figure 5(a) depicts the distribution of the outbound links in the entire trace whereas Fig. 5(b) corresponds to the distribution of the textual size for the visited web pages. It can be seen that a big portion of the trace contains web pages with quite few outbound links. Furthermore, a relative big portion, nearly 46%, contains web pages with less than 200 words. Moreover, in Fig. 6(a) one can see the distribution of the number of bigrams in each web page of the trace after the preprocessing steps described in detail in subsections III-A and III-C. Finally, Fig. 6(b) depicts the distribution of bigrams assigned to each of the outbound links. These bigrams correspond both to linked as-well-as anchored text around the outbound links.

After the processing of the trace, an individual user profile was created for each of the volunteers 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

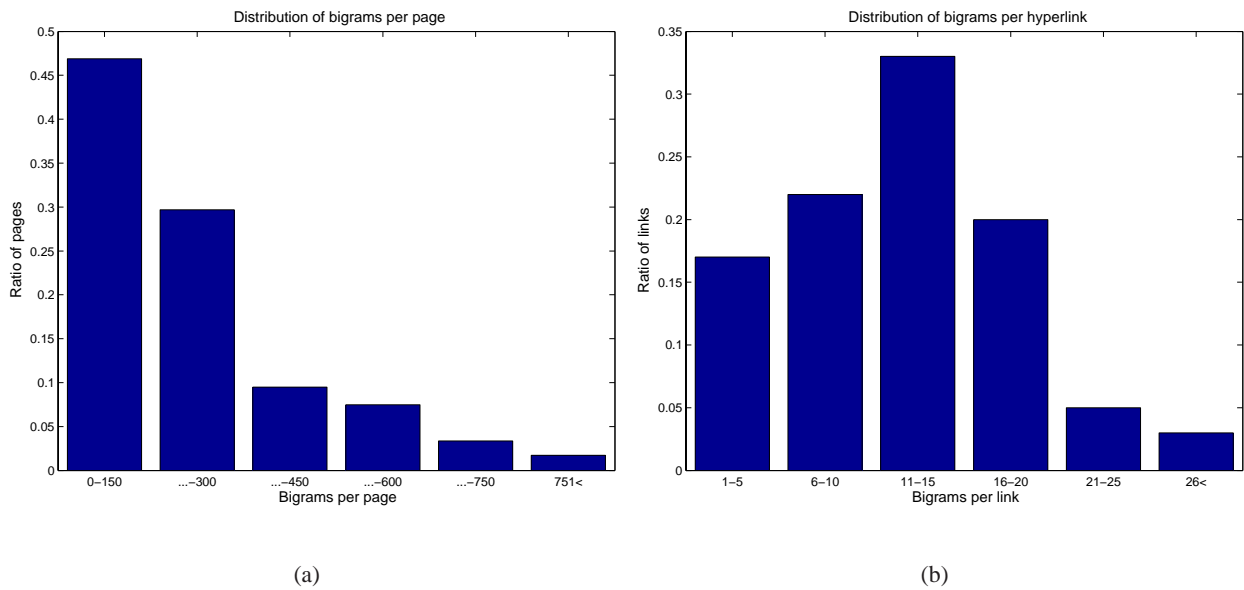


Fig. 6. (a) The number of bigrams per page after the preprocessing steps. (b) The number of bigrams per outbound link.

instructed to follow their normal every-day routine regarding their surfing habits. The profile of each particular volunteer was used by the stand-alone application in order to prefetch a portion of the outbound links for each of the web pages that this particular user visited.

B. Testing set

During the testing phase, the performance of the algorithms was evaluated using two different test sessions. In these sessions two access logs were captured. The logs consists of 82500 and 125000 individual web pages respectively. The first test trace comprises of a 2 month of Internet activity for the same volunteers and it is employed in the performance evaluation that relied on the *recall-precision* ratio. The second trace spanned over a period of nearly 4 months and is employed in a performance evaluation that relies on the *cache-hit* ratio, the *usefulness* of prediction, the *fractional latency reduction* and finally the *fractional network traffic* that the prefetching algorithms may caused to the network.

An important note here is that the contextual statistics that were accumulated during the training phase of the experiment were the starting point for the proposed prefetching algorithm, for each volunteer individually, during the testing phase. Moreover, these statistics were constantly updated during the testing phase for each user separately. That is, each time a user requested a new web page, the steps described

PERFORMANCE METRICS.

Metric name	Symbol	Definition
Hit rate	H	$\frac{\text{num.useful prefetches}}{\text{num.total prefetches}}$
Usefulness	U	$\frac{\text{num.useful prefetches}}{\text{num.total requests}}$
Fractional latency reduction	L_r	$\frac{UPL_{np} - UPL_p}{UPL_{np}}$
Fractional network traffic	T_r	$\frac{\text{num.bytes transmitted}}{\text{num.bytes requested}}$

in subsections III-A and III-C were followed in order to keep the profiles constantly updated with the actual user's preferences. In what follows, subsection V-C provides further details on the performance evaluation that is based on the cache-hit ratio and the rest of the metrics mentioned at the end of the previous paragraph whereas subsection V-D will provide the recall-precision ratios.

C. Hit ratios and network overloads

The performance metrics covered in this subsection are the cache-hit ratio, the usefulness of prediction, the fractional latency reduction and the fractional network traffic [Dav02]. Table II summarizes the metrics that will be used in the current subsection of the performance evaluation.

Cache-hit ratio is the ratio of prefetched pages that a user requested to all the pages the prefetch agent retrieved and represents the accuracy of the prediction. The above metric constitute the accuracy of the prediction. If the hit ratio is high, then the UPL will be low, since almost all requests are served by the local cache. A direct side-effect of the above approach is that if and when a prefetching algorithm is very successful in predicting the forthcoming actions, the overall bandwidth overhead might be disappointing.

Usefulness of predictions is the quantitative relation between the useful prefetched pages (that the user requested) to all the requested pages. Literally, it corresponds to the coverage of the prediction.

Fractional latency reduction is the ratio between the decrease due to prefetching of the observed UPL without a caching system (UPL_{np}) from the UPL with a caching system (UPL_p) to the observed UPL without caching (UPL_{np}). It is important here to describe the mechanism applied for the computation

of the latency reduction. Since a prefetching algorithm will request, in theory, different documents that might be stored in different web servers, a mechanism is needed to detect and obviate any infrequent and abnormal delays that might affect and degrade the overall performance. Therefore, each document is requested from the corresponding web server a multiple number of times during a period of several days and in different time instances during each and every day. Afterwards, a median value is computed which represents the “average” response and completion time for the couple $\langle \text{requested document, hosting web server} \rangle$. Through this process, any isolated abnormalities that might affect the evaluation process are eliminated. Finally, before proceeding to the last metric to be employed, we should note that the median operator was used instead of the mean value due to its robustness properties against outlier observations.

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. It represents the bandwidth overhead added to the network traffic of the non prefetched case, when prefetching is engaged. It is obvious that since some of the future behavior will not be predicted precisely some of the prefetched documents will never be requested. These redundant documents add undesired network traffic and should not have been prefetched.

Any prefetching algorithm should aim to balance the cache-hit ratio and the usefulness against the bandwidth overhead when anticipating future requests. A very strict approach that almost never prefetches documents will definitely keep the anticipated overhead low, but will not benefit the user either. If, on the other hand, we decide to prefetch a plethora of documents in anticipation of capturing the users next request, then we will definitely have to pay high bandwidth cost.

Figure 7(a) depicts the cache-hit ratio curves for each of the four prefetching algorithms. The proposed algorithm corresponds to the curve labeled as “*Bigrams in anchored text*”. The in question figure depicts the average cache-hit curves for all the volunteers in predefined steps of the outbound links volume. It is important to note that interpolation has been applied during the formation of these curves when it was needed. The necessity for interpolation is justified in the cases when the percentage of outbound links in a page is not an integer number. That is, when a page has for example nine outbound links, then prefetching

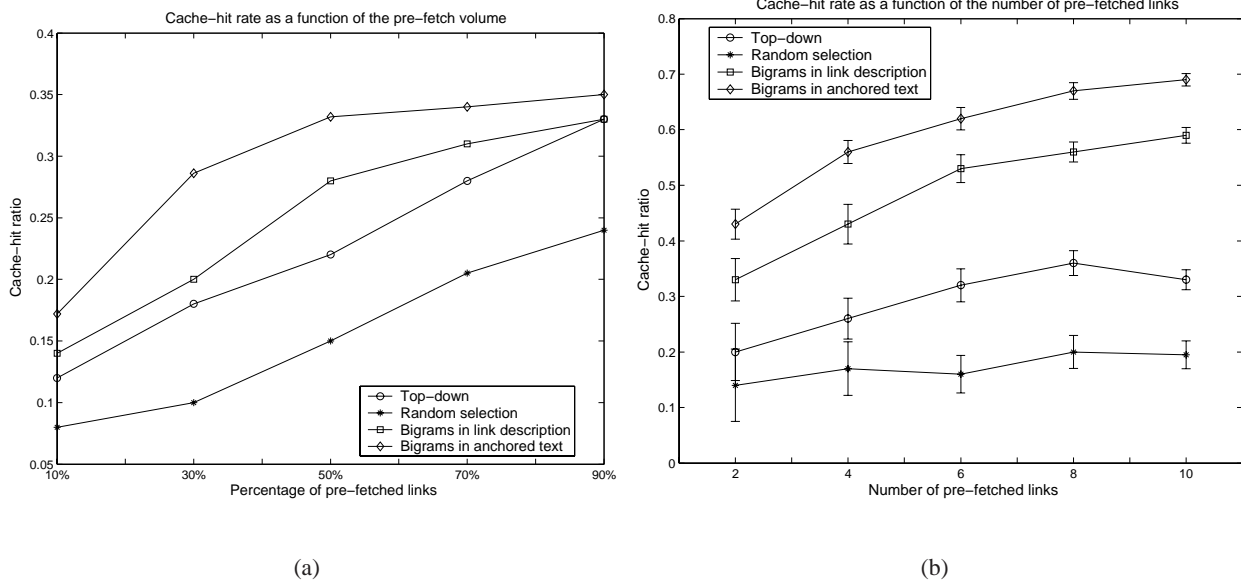


Fig. 7. (a) The average cache-hit ratio curves for each of the four prefetching algorithms in predefined step of the volume of prefetched outbound links. (b) The average cache-hit ratio curves for the prefetching algorithms for a given number of prefetched outbound links.

30% of the links implies 2.7 prefetches which is unfeasible. In that case, the cache-hit ratio was calculated by interpolating the closest values available. From the figure it is evident that all curves will converge to the same cache-hit point when all the outbound links will be prefetched. This is inevitable since the number of useful outbound links is not a function of the prefetching mechanism employed, rather than the personal preferences of each user. Furthermore, the proposed algorithm exhibits better performance than its prime candidate, the “Bigrams in link description”, in each prefetch volume, with a maximum difference of 11.2% higher cache-hit rate which is achieved at a 30% prefetch volume.

Moreover, in Fig. 7(b) one can see a comparison between the algorithms that is based on the number of outbound links that were prefetched. While the previous figure presents an overall evaluation in each prefetch level (up until 90% of the volume of outbound links) the later figure is more concise and covers only a fraction of the outbound links (until ten prefetched links per page). Again the proposed algorithm exhibits better characteristics than its prime candidate but at this time the difference is on average 10.6%. Another important finding is the high cache-hit rate that both algorithms, which rely on textual information, exhibits. This is due to the distribution of the outbound links in the testing trace. In the same figure it can be seen the deviations in each point of the curves. From the deviation marks (vertical lines) it is evident

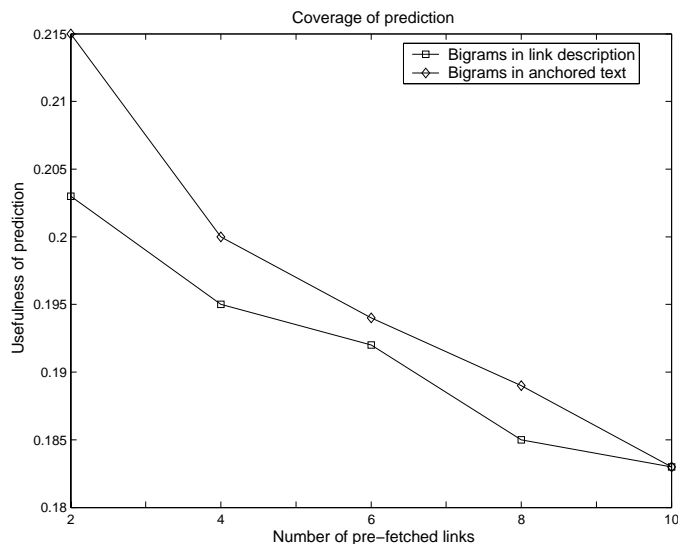


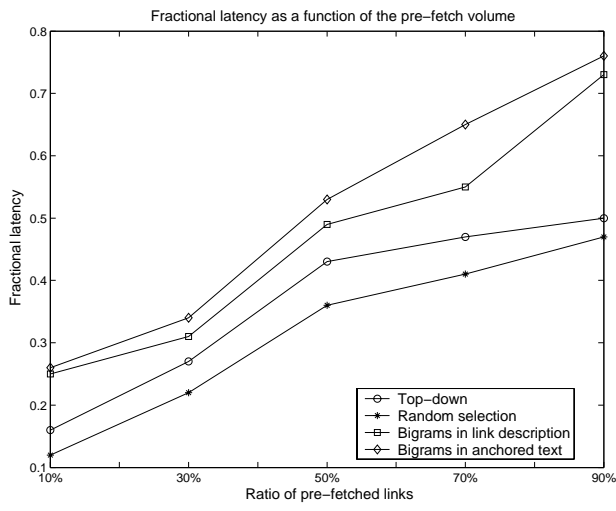
Fig. 8. The usefulness of prediction curves for the prefetching algorithms in predefined steps of the volume of outbound links.

that the prefetching algorithms that rely on textual data present superior performance compared to the other two algorithms namely, the random selection and the top-down approach.

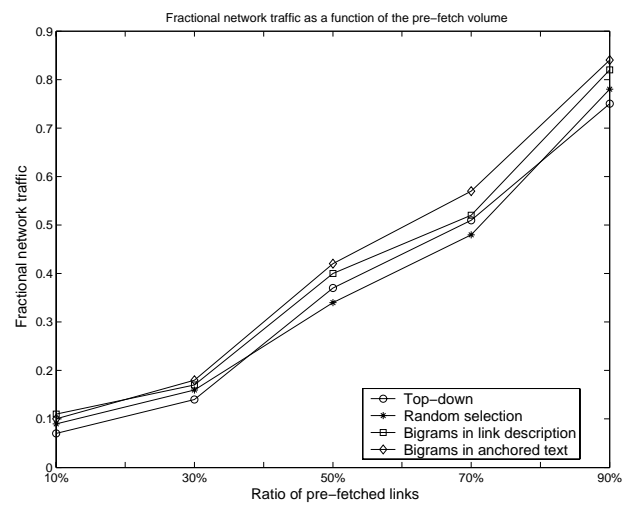
Fig. 8 depicts the usefulness of the prediction. From this figure is apparent that the significant increase of hit ratio is induced with less significant cost in the usefulness. Finally, as it can be seen in Fig. 9(a), prefetching does reduce network latency in all prefetch volumes whereas the bandwidth overhead is more or less the same for all the algorithms (Fig. 9(b)). The latency is reduced from around 25% when only 10% of the links are retrieved to nearly 75% when 90% of the links have been prefetched. 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 1% to 10%.

D. Recall-precision ratio

The performance of the proposed algorithm was assessed also by further employing user input with respect to the accuracy of the prediction of the next requests. The users were asked to decide whether the prefetched documents were relevant to their preferences or not. In doing so the users had to label either as relevant or non-relevant to their preferences each of the outbound links. It should be noted that the input from the user's was asked after the completion of each prefetching session, that is, after finishing



(a)



(b)

Fig. 9. (a) The fractional latency reduction ratio for given volumes of prefetched outbound links. (b) The fractional network traffic ratio for specified volumes of prefetched outbound links.

a particular session for a specific page the user was asked to assign labels to each of the outbound links.

For each user and for each page that was evaluated during the *testing phase* the accompanying procedure was followed: let $\mathcal{L}_s = \{l_1, l_2, \dots, l_{n_s}\}$ denote the set of outbound links that are found in the s th visited document where the indices correspond to the order of appearance of the links inside the web documents. Some of the outbound links are *relevant* to the user's interest and some are *non-relevant*. The relevant links are more probable to be requested and therefore are prime candidates for prefetching.

Let J_i denote the indicator:

$$J_i = \begin{cases} \text{true,} & \text{the } i\text{th link is relevant to the user} \\ \text{false,} & \text{the } i\text{th link is not relevant to the user} \end{cases}. \quad (12)$$

So for each set \mathcal{L}_s there is a vector, \mathcal{F}_s , containing the indicators that correspond to the user's opinion on the relevance or not of the links.

Each prefetching algorithms available today actually performs a permutation of the elements of the vector \mathcal{F}_s and subsequently retrieves a predetermined subset of links. For each page viewed by the user (*i.e.* the s th web page), the retrieval of a portion of the outbound links leads to a partitioning of \mathcal{L}_s into two disjoint sets, \mathcal{L}_s^m , where $m \in \{0, 1\}$. The set \mathcal{L}_s^0 contains the non-relevant links and \mathcal{L}_s^1 contains the

CONTINGENCY TABLE FOR PREFETCHING EVALUATION.

	<i>Retrieved</i>	<i>Not-Retrieved</i>	
<i>Relevant</i>	r	x	$n_1 = r + x$
<i>Not Relevant</i>	y	z	
	$n_2 = r + y$		

relevant links respectively.

Table III is the 2×2 contingency table which shows how the collection of outbound links is divided. In the contingency table, n_1 corresponds to the total number of relevant links and n_2 denotes the number of retrieved pages [Kor97]. Then, the effectiveness of an algorithm can be measured using the so-called *precision* and *recall* ratios. Precision is defined as the proportion of retrieved documents that are relevant, $P = \frac{r}{n_2}$, whereas, recall is the proportion of relevant documents that are retrieved, $R = \frac{r}{n_1}$.

During the testing phase of the algorithms the user is requested to validate as either relevant or non-relevant the outbound links in each visited page. Then, for each one of the above mentioned algorithms we compute the recall-precision ratios. As the volume of retrieved documents increases the above ratios are expected to change. The sequence of $\langle \text{recall}, \text{precision} \rangle$ pairs obtained yields the so-called *recall-precision* curve. An average over all the curves corresponding to the visited web pages produces the *eleven point average* recall-precision curve [Kor97]. The comparison of the effectiveness between the four prefetching algorithm utilizes the above-mentioned curve. Figure 10 depicts the average recall-precision curves for all the prefetching schemes. It can also be seen in Fig. 10 that the proposed algorithm performs better than the rest of the prefetching techniques in the entire range of recall volumes.

E. Statistical evaluation

To assess whether the observed differences in the recall-precision ratios collected during the evaluation between the proposed algorithm and the rest of the algorithms are really meaningful or merely a chance, a hypothesis test is employed. Several hypothesis tests can be used that range from the t -test and the signed

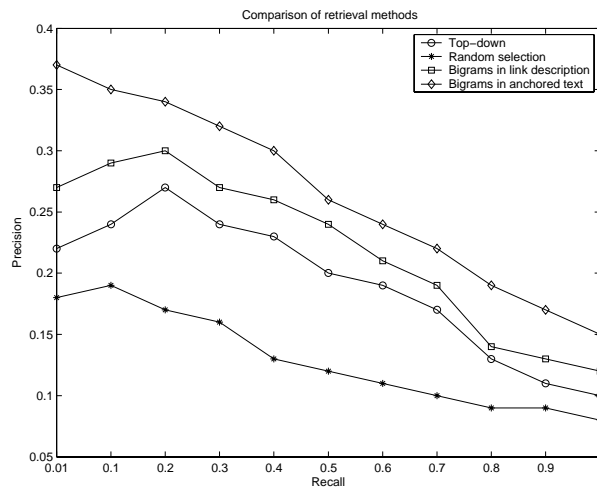


Fig. 10. The recall precision curves for each one of the four algorithms.

rank test to the Wilcoxon test and the one-way ANOVA test when comparing between two different algorithms [Hul93]. For comparing more than two algorithms simultaneously, the two-way ANOVA test and the Friedman test can be employed [Seb02], [YL99].

Let us denote by P_{ij} the i th precision score of the j th prefetching algorithm, where $j = 1, 2, \dots, m$ and $i = 1, 2, \dots, n$. To compare the performance between these algorithms, the two-way analysis of variance (ANOVA) will be employed. Let \bar{P}_j represents the average precision score for the j th algorithm, \bar{P}_i represents the average recall score for all the algorithms for the i th recall level and \bar{P} represents the overall average. The ANOVA test relies on the following statistic:

$$F_A = \frac{n(n-1) \sum_{j=1}^m (\bar{P}_j - \bar{P})^2}{\sum_{j=1}^m \sum_{i=1}^n (P_{ij} - \bar{P}_j - \bar{P}_i + \bar{P})^2}, \quad (13)$$

which follows the F distribution with $(m-1)$ and $(m-1)(n-1)$ degrees of freedom respectively. The null hypothesis to be tested is that the mean precision for each algorithm is statistically equal under the alternative hypothesis which is the negation of H_0 . If H_1 is accepted, then at least one of the algorithms under consideration exhibit mean precision statistically significant than the rest of the algorithms. In that case a pairwise tests for differences in the average precisions must be carried out.

The F_A value computed for the four algorithms is 9.42 and the p-value for the null hypothesis which

TWO SAMPLE t -TEST BETWEEN THE FOUR ALGORITHMS.

Pair of algorithms	Reject H_0	
	0.1	0.05
<i>top-down</i> - proposed	true	true
<i>random selection</i> - proposed	true	true
<i>bigrams in link description</i> - proposed	false	false

states that all the algorithms derive from the same population or from different populations with the same mean precision value is 7.8150×10^{-5} . This lead to the assumption that for at least one of the algorithms the mean value is significantly different. The box and whisker plot in Fig. 11(a) confirms the above findings. We can clearly see that the mean precision value of the suggested algorithm exceeds the other three algorithms. Furthermore, Figure 11(b) depicts the confidence intervals for the mean precision values of the four prefetching algorithms. It is clear that the suggested algorithm performs better than the rest of the algorithms and there is just a slight superposition in the confidence intervals of the suggested algorithm and the algorithm that uses the bigrams in the link description.

In order to verify the superiority of the proposed algorithm we will perform two sample t -test for each pair of algorithms. The null hypothesis for the two sample t -test is that the algorithms being tested are not significantly different. Table IV depicts the results of the hypothesis test for H_0 under the 0.1 and 0.05 levels of significance for each pair of algorithms. It is evident that for the pairs $\langle \textit{top-down}, \textit{proposed} \rangle$ and $\langle \textit{random selection}, \textit{proposed} \rangle$ the null hypothesis can be rejected, that is, the proposed algorithm is statistically significant for both levels of significance. On the other hand the pair $\langle \textit{bigrams in link description}, \textit{proposed} \rangle$ exhibits the same characteristics. The significance value for the t -test in this case is 0.16.

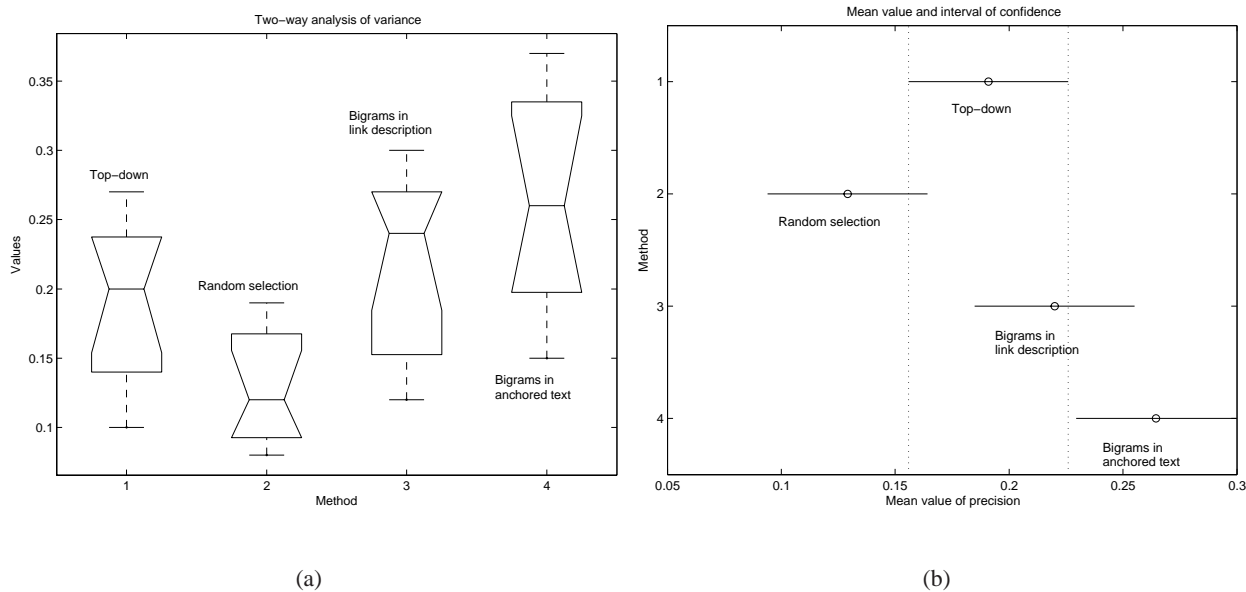


Fig. 11. (a) The box and whisker plot of the two-way analysis of variance between the four tested algorithms. (b) Confidence intervals for the mean precision values of the four prefetching algorithms.

VI. CONCLUSIONS

In this paper we provided a novel transparent and speculative algorithm to model the user's behavior when surfing the Internet. The proposed modeling algorithm is based on the frequency of occurrence for selected bigrams forming the visited web pages. 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 algorithm prefetches some of the linked documents and stores them locally in a cache in an effort to reduce the UPL. The proposed algorithm has been tested against three 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

- [AWY99] C. C. Aggarwal, J. Wolf, and P. S. Yu. Caching on the world wide web. *IEEE Trans. on Knowledge and Data Eng.*, 11(1):95–107, 1999.
- [Bes95] A. Bestavros. Using speculation to reduce server load and service time on the WWW. In *Proc. of 4th ACM Int. Conf. on Information and Knowledge Management (CIKM95)*, pages 403–410, November 1995.

- [BP98] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.
- [BS00] B. Berendt and M. Spiliopoulou. Analysis of navigation behavior in web sites integrating multiple information systems. *The VLDB Journal*, 9(1):56–75, 2000.
- [Cha99] P. K. Chan. A non-invasive learning approach to building web user profiles. In *Proc. of WebKDD’99*, pages 7–12, August 1999.
- [CK00] E. Cohen and H. Kaplan. Prefetching the means for document transfer: A new approach for reducing Web latency. In *Proc. of IEEE INFOCOM*, 2000.
- [CKR98] E. Cohen, B. Krishnamurthy, , and J. Rexford. Improving end-to-end performance of the web using server volumes and proxy filters. In *Proc. of SIGCOMM98*, pages 241–253, 1998.
- [Cor02a] Deerfield Corporation. Interquick for wingate. "http://interquick.deerfield.com", 2002.
- [Cor02b] Goto Software Corporation. Webearly3 product page. "http://www.goto-software.com/WE3/WEstart.htm", 2002.
- [CY97] K. Chinen and S. Yamaguchi. An interactive prefetching proxy server for improvement of www latency. In *Proc. of INET’97*, 1997.
- [Dav02] B. D. Davison. *The design and evaluation of web prefetching and caching techniques*. PhD thesis, State University of New Jersey, 2002.
- [DK01] M. Deshpande and G. Karypis. Selective markov models for predicting web-page accesses. In *Proc. of SIAM Int. Conf. on Data Mining (SDM’01)*, 2001.
- [Duc99] D. Duchamp. Prefetching hyperlinks. In *Proc. of 2nd USENIX Symposium on Internet Technologies and Systems (USITS’99)*, October 1999.
- [FB91] N. Fuhr and C. Buckley. A probabilistic learning approach for document indexing. *ACM Trans. on Information Systems*, 9(3):223–248, 1991.
- [FCJ99] L. Fan, P. Cao, and Q. Jacobson. Web prefetching between low-bandwidth clients and proxies: Potential and performance. In *Proc. of Joint Int. Conf. on Measurement and Modeling of Computer Systems (SIGMETRICS99)*, pages 178–187, 1999.
- [FS00] D. Foygel and D. Strelow. Reducing web latency with hierarchical cache based prefetching. In *Proc. of Int. Workshop on Scalable Web Services*, pages 103–110, 2000.
- [GH03] M. Géry and H. Haddad. Evaluation of web usage mining approaches for users next request prediction. In *Proc. of 5th Int. Workshop on Web Information and Data Management (WIDM’05)*, pages 74–81, 2003.
- [GST02] W. Gaul and L. Schmidt-Thieme. Recommender systems based on user navigational behavior in the internet. *Behaviormetrika*, 29:1–22, 2002.
- [Hul93] D. Hull. Using statistical testing in the evaluation of retrieval performance. In *Proc. of 16th ACM/SIGIR Int. Conf. on Research and Development in Information Retrieval*, pages 329–338, 1993.

- [Inc02] CacheFlow Inc. Active caching technology. "http://www.cacheflow.com/technology/whitepapers/active.cfm", 2002.
- [Kle99] R. P. Klemm. WebCompanion: A friendly client-side Web prefetching agent. *IEEE Trans. on Knowledge and Data Eng.*, 11(4), 577–594 1999.
- [Kor97] R. R. Korfhage. *Information Storage and Retrieval*. NY: J. Wiley, 1997.
- [KT01] J. I. Khan and Q. Tao. Partial prefetch for faster surfing in composite hypermedia. In *Proc. of 3rd USENIX Symposium on Internet Technologies (USITS01)*, pages 13–24, 2001.
- [KW98] A. Kraiss and G. Weikum. Integrated document caching and prefetching in storage hierarchies based on markov-chain predictions. *The VLDB Journal*, 1998.
- [MC98] E. P. Markatos and C. Chronaki. A top-10 approach to prefetching on the Web. In *Proc. of INET'98*, 1998.
- [MDLN01] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Effective personalization based on association rule discovery from web usage data. In *Proc. of 3rd Int. Workshop on Web Information and Data Management (WIDM'01)*, pages 9–15, 2001.
- [Mla98] D. Mladenić. Feature subset selection in text learning. In *Proc. of 10th European Conf. on Machine Learning (ECML'98)*, pages 95–100, 1998.
- [MS99] D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press, 1999.
- [PHMAZ00] J. Pei, J. Han, B. Mortazavi-Asl, and H. Zhu. Mining access patterns efficiently from web logs. In *Proc. of Pacific-Asia Conf. in Knowledge Discovery and Data Mining (PAKDD00)*, 2000.
- [PM96] V. N. Padmanabhan and J. C. Mogul. Using predictive prefetching to improve world wide web latency. *Computer Communication Review*, 26(3):22–36, 1996.
- [PM99] T. Palpanas and A. Mendelzon. Web prefetching using partial match prediction. In *Proc. of Web Caching Workshop*, 1999.
- [Por80] M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
- [PP99a] P. L. Pirolli and J. E. Pitkow. Distributions of surfers' paths through the World Wide Web: Empirical characterization. *World Wide Web*, 2(1–2):29–45, 1999.
- [PP99b] J. E. Pitkow and P. L. Pirolli. Mining longest repeated subsequences to predict World Wide Web surfing. In *Proc. of 2nd USENIX Symposium on Internet Technologies and Systems (USITS'99)*, 1999.
- [Res99] Zona Research. The economic impacts of unacceptable web-site download speeds, April 1999.
- [Ros00] R. Rosenfeld. Two decades of statistical language modeling:where do we go from here? *Proceedings of the IEEE*, 83(8):1137–1380, 2000.
- [RS99] M. E. Ruiz and P. Srinivasan. Hierarchical neural networks for text categorization. In *Proc. of 22nd ACM/SIGIR Int. Conf. on Research and Development in Information Retrieval*, pages 281–282, 1999.
- [Sar00] R. R. Sarukkai. Link prediction and path analysis using markov chains. In *Proc. of 19th Int. World Wide Web Conf.*, 2000.
- [Seb02] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

- [SSV99] J. Shim, P. Scheuermann, and R. Vingralek. Proxy cache algorithms: Design, implementation, and performance. *IEEE Trans. on Knowledge and Data Eng.*, 11(4):549–562, 1999.
- [SSV00] F. Sebastiani, A. Sperduti, and N. Valdambrini. An improved boosting algorithm and its application to automated text categorization. In *Proc. of 9th ACM Int. Conf. on Information and Knowledge Management*, pages 78–85, 2000.
- [Wan99] J. Wang. A survey of web caching schemes for the internet. *ACM Computer Communication Review*, 29(5):36–46, October 1999.
- [WC96] Z. Wang and J. Crowcroft. Prefetching in world wide web. In *Proc. of IEEE Global Internet*, pages 28–32, 1996.
- [YL99] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proc. of 22nd ACM/SIGIR Int. Conf. on Research and Development in Information Retrieval*, pages 42–49, 1999.
- [YP97] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proc. of 14th Int. Conf. on Machine Learning (ICML'97)*, pages 412–420, 1997.
- [YSG02] Y. Yang, S. Slattery, and R. Ghani. A study of approaches to hypertext categorization. *Intelligent Information Systems*, 18(2/3):219–241, 2002.
- [YZ03] Q. Yang and H. H. Zhang. Web-log mining for predictive web caching. *IEEE Trans. on Knowledge and Data Eng.*, 15(4):1050–1053, 2003.
- [YZL01] Q. Yang, H. H. Zhang, and I. T. Y. Li. Mining Web logs for prediction models in WWW caching and prefetching. In *Proc. of 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 473–478, 2001.