

BEHAVIOR MODELING USING BIGRAM FREQUENCIES FOR CLIENT-SIDE LINK PREFETCHING

A. Georgakis* and H. Li
Digital Media Laboratory (DML),
Department of Applied Physics and Electronics,
Umeå University, SE-90187 Umeå Sweden
{apostolos.georgakis, haibo.li}@tfe.umu.se

M. Gordan
Basis of Electronics Department,
Technical University of Cluj-Napoca,
Cluj-Napoca, RO-3400, Romania
mihag@bel.utcluj.ro

ABSTRACT

The perceived latency for a user surfing the Internet is the target of a transparent and speculative algorithm that relies on a user behavior model. The model is based on past user behavior and in combination with a weighting scheme for the outbound links of a particular web page, aims at reducing the perceived latencies. The assistance is in the form of prefetching some linked web pages and storing them in the browser's cache. A comparison between the proposed algorithms against two other prefetching algorithms yield improved cache-hit rates given a moderate bandwidth overhead. Furthermore, the experimental results are proven to be statistically significant.

KEY WORDS

Pre-fetching, user behaviour modelling, bigrams

1 Introduction

Thanks to the ever growing popularity of the *World Wide Web* and the familiarization of the average users with this new medium as well as the ever increasing use of multimedia content the *User Perceived Latencies* (UPL) for web pages hosted in a plethora of the web servers are extremely high. The time it takes to download a particular document can be increased furthermore when the user's *Internet Service Provider* (ISP) performance is added in the above equation. In that case, UPL becomes also a product of the connection speed from both the user's ISP and the respective backbone provider.

It is obvious that a decent response time have positive effects on both the users and the web sites. The satisfaction perceived due to the productivity increase for the end-user is reflected back to the web site. A rather old but still useful study conducted in 1999 by Zona Research Inc. provides evidence that if a site takes more than eight seconds to load then the user is much more likely to search for an alternative source provider [1]. The above time frame might seem far-

etched but intuition says that there should definitely be a time threshold above which an average user will give up waiting for the document to be downloaded and move to other directions.

A plethora of solutions have been suggested and are currently implemented in an effort to decrease the UPL. The solutions cover a wide range of fast connections (xDSL, LAN, cable modem etc) or alternative communication technologies (satellite, wireless, radio LANs etc). Moreover, fast servers in hardware and software level, increased bandwidth either from a single backbone provider or from multiple providers and usage of proxy servers by the ISP for local caching of frequent or "popular" documents. Finally, server side proxies and distributed servers for traffic balancing (bandwidth management using multiple servers) are also used by popular web sites, which can substantially improve typical response times. Still, the latencies exist and are unfortunately amplified when the user is visiting web pages and documents that has never visited in the past.

Another possibility for retrieval latency reduction without hardware modifications and extra costs for the end-user can come from the usage of the browser's cache [2, 3]. In this approach the documents that have higher probability to be accessed are *prefetched* and stored locally in the browser's cache. The next requests for one of these documents is served instantly since the document already exist locally thus avoiding any unnecessary connection to some remote server. Locally serving the document avoids all the delays previously mentioned.

Naturally a difficulty that arise here is to accurately determine the forthcoming requests in order to prefetch these documents into a local cache. This is in order to raise the probability of correctly capturing the users interest. In that case some prefetched documents may and will never be used resulting in a waste of bandwidth and an increased workload for the servers. Therefore, the constraint to be met is to determine as accurate as possible the user's interests in order to download only the documents that are of the user's interest and reduce the erroneous downloadings.

In this paper we will present an algorithm that

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

models the user's past Internet behavior. The proposed algorithm creates a user profile using the frequency of occurrence of a selected subset of the bigrams forming the visited web pages. These frequencies are then used in conjunction with the text anchored around the outbound links to compute weights for these links. Subsequently, the algorithm retrieves the linked documents according to the computed weights.

In what follows, Section 2 provides the needed details for the preprocessing of the web pages, the evaluation of the contextual statistics for the bigrams forming the visited web pages, the creation of the user's profile, the prediction scheme which assigns weights to the outbound links and finally, the usage of user input in some stage of the algorithm. Section 3 provides the experimental results which are based on the cache-hit ratio, the usefulness of prediction, the fractional latency reduction and the fractional network traffic. Finally, Section 4 provides a statistical evaluation of the experimental results.

2 Proposed Algorithm

This paper provides a novel *transparent* and *speculative* prefetching algorithm for Web documents on behalf of the user. By "transparent" we mean that the user is not involved in the selection of the documents to be prefetched and "speculative" means that the algorithm "predicts" the user's future requests based on knowledge acquainted during past references. Therefore, a user profile needs to be created based on the user's *past behavior*. In doing so it is assumed that there is a correlation between the textual content of the web pages viewed by the user and the particular user's interests. Subsequently, the textual information extracted from the visited pages is stored in a profile and used in order to decide which of the documents linked on the currently viewed page to be retrieved for probable latter request.

The correlation between content and user preferences is unknown but there is no *non-invasive* and *direct* method to assess the degree of that correlation. Therefore we need to rely on *indirect* approaches to filter out the non-relative content. This chapter will provide a plethora of tools used in that direction.

2.1 Preprocessing

The first step toward 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.
- 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 [4].

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 the above mentioned preprocessing steps the resulted web page contains 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 take into consideration the so-called *low-to-medium* "law" from the Information Retrieval community. According to this law the terms whose document frequency is low-to-medium are the most informative ones [5, 6]. The above law implies that the stems with low document frequency are not informative regarding the actual content of the documents; they rather help in the formation of the text inside the sentences and the pages and therefore can be eliminated. In the case under consideration the high-frequency stems whose cumulative probability equals 10% of the total probability mass as well as the low-frequency stems with the same cumulative probability are eliminated.

2.2 Contextual statistics evaluation

For the construction of the user's profile the algorithm will evaluate the contextual statistics of the stems that remained after the preprocessing step. A well-know modeling method from the *statistical language modeling* community will be used here [6]. Let $\mathcal{S} = \{s_i\}$

denote the set of stems (vocabulary) found in the collection of documents, where $i \in \{1, 2, \dots, |\mathcal{S}|\}$ and $|\cdot|$ denotes the cardinality of a set. The *a priori* probability $P(s_1^* s_2^* \dots s_m^*)$ of the sequence $s_1^* s_2^* \dots s_m^*$ with $s_i^* \in \mathcal{S}$ can be expressed as the product of conditional probabilities $P(s_1^* s_2^* \dots s_m^*) = \prod_{i=1}^m P(s_i^* | h_i)$ where $h_i = \{s_1^* s_2^* \dots s_{i-1}^*\}$ denotes the *history* of the i th stem.

In this paper, we adopt the *bigram* model where only one preceding stem is used to encode each stem, *i.e.*:

$$P(s_i | h_i) \simeq P(s_i | s_{i-1}) = \frac{\eta(s_{i-1}, s_i)}{\eta(s_{i-1})}, \quad (1)$$

with $\eta(s_{i-1}, s_i)$ and $\eta(s_{i-1})$ denoting the number of observed bigrams (s_{i-1}, s_i) and stem types s_{i-1} , respectively, in the collection of processed documents.

2.3 Profile creation

Let $A_{N,M}$ denote the bigram-to-document indicator matrix, that is, $A_{N,M} = (a_{ij})$ where N denotes the number of bigrams seen by the algorithm so far and M denotes the number of documents visited by the user. Furthermore, a_{ij} is a boolean valued variable where $a_{ij} = 1$ when the i th bigram is present in the j th document and $a_{ij} = 0$ when the bigram is absent. It must be noted here that $A_{N,M}$ comprises of the documents that the user have visited and not the documents that were selected for prefetching by the algorithm. Let also $\mathcal{F} = (f_{b2d}(b_1), f_{b2d}(b_2), \dots, f_{b2d}(b_N))$ denote the global bigram-to-document frequencies, where $f_{b2d}(b_i) = \sum_{k=1}^M a_{ik}$, that is the number of documents in which the i th bigram is present at least once. Obviously, high bigram-to-document frequencies signify important bigrams since they are found nearly in all the documents visited by the user. Therefore, these bigrams with high document frequency will be used to model the user behavior. By thresholding the low frequent bigrams (grayed area) the algorithm forms the user profile.

2.4 Prediction algorithm

The anticipation of future actions based on a model of past behavior has gained much attention in recent years. Such a model can be build either in active mode where the user provides input and builds a personal profile or in passive mode where the user's behavior is observed but no input is required [7]. The passive mode of creation which was described in the subsection 2.3 will be used for the prediction of future actions and its prediction accuracy will be enhanced by using limited user input.

Let \mathcal{B} denote the set of bigrams comprising the user's profile, that is, $\mathcal{B} = \{b_1, b_2, \dots, b_{|\mathcal{B}|}\}$. Let also \mathcal{X}_s denote the bag of bigrams, $\mathcal{X}_s = \{b_1^s, b_2^s, \dots, b_{n_s}^s\}$,

associated with the s th outbound link in the page currently displayed by the browser where n_s corresponds to the number of bigrams in this particular document. In order to assign a weight to a particular link the proposed algorithm will take into consideration the fact that the frequency of appearance for a given bigram denotes the bigram's importance for the user. So, by adding the frequencies of the bigrams comprising a particular link the algorithm can get a rough approximation of its importance. The previous is transcribed into :

$$w_1^s = \frac{\sum_{i=1}^{n_s} f_{b2d}(b_i^s)}{n_s}. \quad (2)$$

Unfortunately, the global bigram-to-document frequency might not reflect the true trends of the user's interest when surfing the net; rather the method of presenting the information contained in the web pages. On the other hand the user's interest is reflected by the path of followed links. In that case the algorithm must exploit this information as well. And since the selection of a link is governed by the text anchored around the particular link, this textual information is a vital sign of the user's preference. In order to grasp the user's behavior, the anchored text around the visited as well as the non-visited links must be taken into consideration. In doing so the algorithm keeps a record of the frequency of appearance for each bigram in the anchored text in both the visited and the non-visited links. That is, for each bigram b_i corresponds two different counters. The first counter which will be denoted by $v_{b2l}(b_i)$ records the frequency of appearance of the bigram in the visited links. The second counter is denoted by $m_{b2l}(b_i)$ and corresponds to the frequency of the links that were not visited by the user. Then the importance of that bigram is:

$$\frac{v_{b2l}(b_i)}{p} - \frac{m_{b2l}(b_i)}{q}, \quad (3)$$

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 found around the links is:

$$w_2^s = \sum_{i=1}^{n_s} \left[\frac{v_{b2l}(b_i^s)}{p} - \frac{m_{b2l}(b_i^s)}{q} \right]. \quad (4)$$

And the weight of the s th link is determined by the following linear combination:

$$T_s = aw_1^s + (1 - a)w_2^s \quad (5)$$

where $a \in \{0, 1\}$ is a scaling parameter. This parameter is adjusted by the user and manages the aggressiveness of the prediction. What this means is that although the content of the visited pages might reveal partially the preferences its the path of visited links

that captures the user’s attention. Therefore, if the user feels confident that the visited links are relevant to the preferences and not just a simple random walk then the parameter a can be set close to unity otherwise it can be set to any other value. By setting the value equal or close to one any random walking in the past has a very little effect in the final performance.

2.5 User input

Since the extraction of a reliable user model or profile is extremely difficult, due to the complexity of natural language, and despite of all the preprocessing steps taken, user input can be used in order to elevate the overall performance of the suggesting weighting algorithm. More specifically, the user can suggest to the algorithm a list of keywords that signify his or her special interest. This list can be enriched with extra keywords using a thesaurus.

The user suggested keywords are used by the algorithm in a way similar to the bigrams in the text anchored around a link. Let $k_i, i = 1, 2, \dots, N_k$ denote the user defined keywords, where N_k is the number of suggested keywords. For each of the keywords the algorithm keeps 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 links. So the importance of the i th keyword is:

$$\frac{\eta_v(k_i)}{p} - \frac{\eta_{nv}(k_i)}{q}. \quad (6)$$

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

$$\max\left(f_{b2d}(b_i^s), \left[\frac{v_{b2l}(b_i^s)}{p} - \frac{m_{b2l}(b_i^s)}{q}\right]\right). \quad (7)$$

So the total weight for the s th link:

$$T_s = aw_1^s + (1-a)w_2^s + \frac{\sum_{i=1}^{N_k} \max(\Omega) \left[\frac{\eta_v(k_i)}{p} - \frac{\eta_{nv}(k_i)}{q}\right] I(k_i, X_s)}{\sum_{i=1}^{N_k} I(k_i, X_s)}, \quad (8)$$

where $I(k_i, X_s)$ is an indicator factor that takes the value 1 if the user supplied keyword is present in the anchored text and $\Omega = \left\{f_{b2d}(b_i^s), \left[\frac{v_{b2l}(b_i^s)}{p} - \frac{m_{b2l}(b_i^s)}{q}\right]\right\}$. It must be noted here that the input from the user is not needed but it always helps elevate the performance of the algorithm.

3 Assessment of the prefetching capabilities

The performance of the proposed algorithm is tested against three different prefetching algorithms which are:

- top-down algorithm
- random permutation of the outbound links
- weighting of the links according to the bigrams forming the *link text* (no usage of the *anchored text around the link*)

In evaluating the performance of the algorithms a trace consisting of nearly 450000 individual web pages was used for training and a trace of 150000 pages was used for testing. 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.

The performance metrics employed are the cache-hit ratio, the usefulness of prediction, the fractional latency reduction and the fractional network traffic [8]. *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. *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}). Finally, *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.

Figure 1(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 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 fetched. 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 12.2% higher cache-hit rate which is achieved at a 30% prefetch volume. Figure 1(b) 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 (maximum observed overhead between the proposed algorithm and its prime candidate is 9.05% when there is an 80% prefetching). Finally, as it can be seen in Fig. 1(c), prefetching does reduce network latency in all prefetch volumes whereas the bandwidth overhead is more or less the same for all the algorithms (Fig. 1(d)). The latency is reduced from around 25% to 30% 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 5% when again compared with the prime candidate which is the “Bigrams in the link description”.

4 Statistical evaluation

To assess whether the observed differences in the evaluation between the proposed algorithm and the other three 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 rank test to the Wilcoxon test and the one-way ANOVA test when comparing between two different algorithms [9]. For comparing more than two algorithms simultaneously, the two-way ANOVA test and the Friedman test can be employed [5, 9, 10].

To compare the performance between these algorithms, the two-way analysis of variance (ANOVA) will be employed. The null hypothesis to be tested is that the mean hit score 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 hit score statistically significant than the rest of the algorithms. In that case a pairwise tests for differences in

Table 1. Two sample t-test - Rejection of H_0

Algorithms	Cache-hit		Net. traffic	
	0.05	0.01	0.05	0.01
1 – 4	true	true	false	false
2 – 4	true	true	false	false
3 – 4	true	false	false	false

the average hit scores must be carried out.

The F_A value computed for the four algorithms is 5.189 and the p-value for the null hypothesis is 1.08×10^{-2} . This lead to the assumption that for at least one of the algorithms the mean value is significantly different. 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. The results of the pair-wise t -test can be seen in Table 1. From the table is evident that statistically the proposed algorithm exhibits superior performance than the rest of the algorithms in both significant levels except in the case of the algorithm termed “Bigrams in link description” for a significance level of 0.01. Furthermore, the null hypothesis can not be rejected for and pair of prefetching schemes when it comes to the fractional network traffic for both significance levels.

In Table 1 and in the first column the association between numbers and algorithms is as follows; 1:top-down algorithm, 2:random permutation of the outbound links, 3: bigrams in link description and 4: bigrams in anchored text.

5 Conclusions

A novel transparent and speculative algorithm was proposed in this paper in an effort to model the user’s behavior when surfing the Internet. The proposed algorithm relies on the frequency of occurrence for selected bigrams forming the visited web pages. The constructed model is used for the prediction of future actions. The proposed algorithm has been tested against three other algorithms and demonstrated superior performance in predicting the user’s future requests which was proven to be statistically significant.

References

- [1] Research, Z.: The economic impacts of unacceptable web-site download speeds (1999)
- [2] Aggarwal, C.C., Wolf, J., Yu, P.S.: Caching on the world wide web. IEEE Trans. on Knowledge and Data Eng. **11** (1999) 95–107

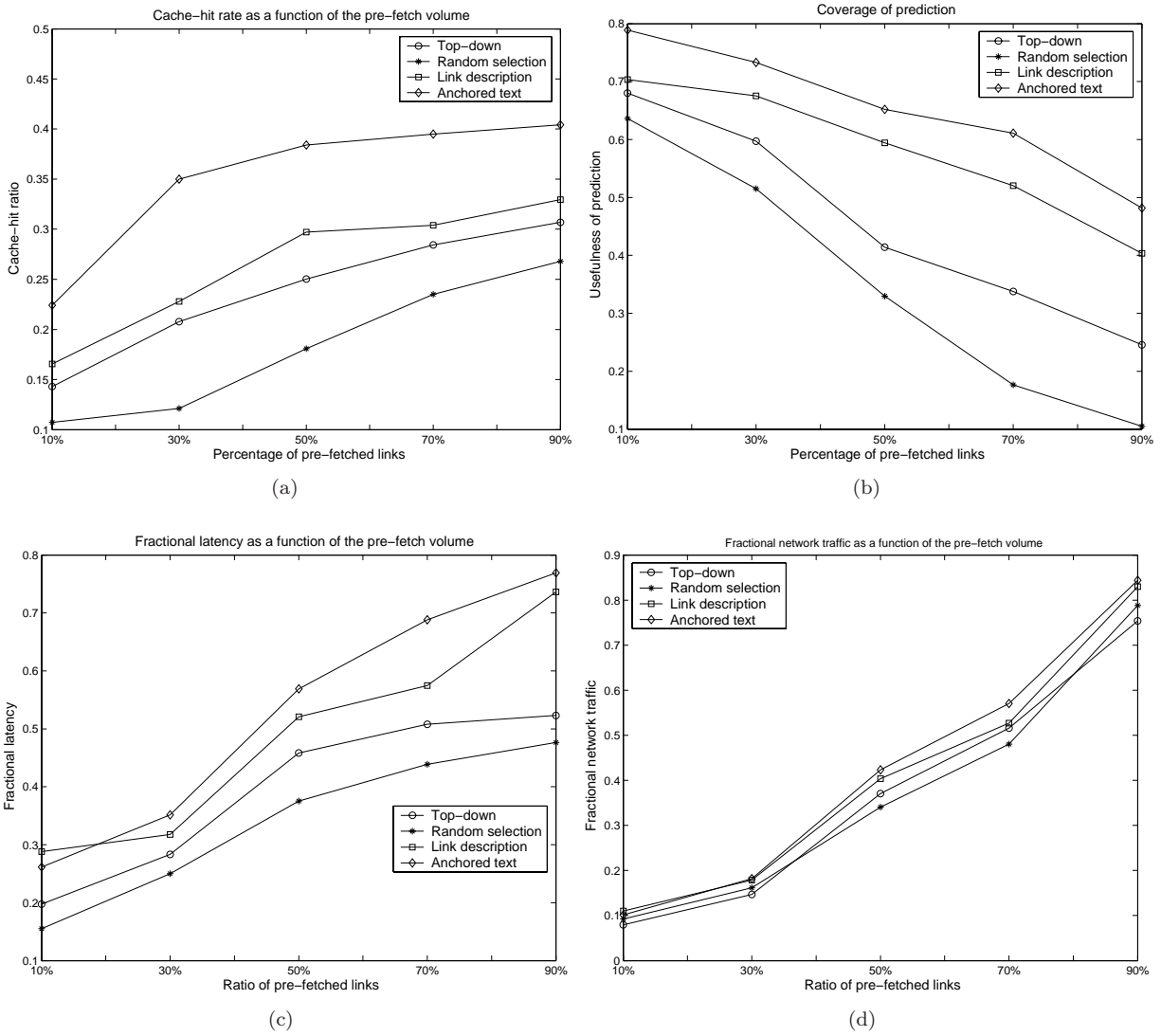


Figure 1. The average performance for each of the four pre-fetching algorithms for all the volunteers and for: (a) cache-hit ratio, (b) usefulness of prediction, (c) fractional latency reduction ratio and (d) fractional network traffic ratio.

[3] Shim, J., Scheuermann, P., Vingralek, R.: Proxy cache algorithms: Design, implementation, and performance. *IEEE Trans. on Knowledge and Data Eng.* **11** (1999) 549–562

[4] Porter, M.: An algorithm for suffix stripping. *Program* **14** (1980) 130–137

[5] Sebastiani, F.: Machine learning in automated text categorization. *ACM Computing Surveys* **34** (2002) 1–47

[6] Manning, D., Schütze, H.: *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press (1999)

[7] Chan, P.K.: A non-invasive learning approach to building web user profiles. In: *Proc. of WebKDD’99*. (1999) 7–12

[8] Davison, B.D.: The design and evaluation of web prefetching and caching techniques. PhD thesis, State University of New Jersey (2002)

[9] Hull, D.: Using statistical testing in the evaluation of retrieval performance. In: *Proc. of 16th ACM/SIGIR Int. Conf. on Research and Development in Information Retrieval*. (1993) 329–338

[10] Yang, Y., Liu, X.: A re-examination of text categorization methods. In: *Proc. of 22nd ACM/SIGIR Int. Conf. on Research and Development in Information Retrieval*. (1999) 42–49