A Keyword-Based Semantic Prefetching Approach in Internet News Services

Cheng-Zhong Xu, *Senior Member*, *IEEE*, and Tamer I. Ibrahim

Abstract—Prefetching is an important technique to reduce the average Web access latency. Existing prefetching methods are based mostly on URL graphs. They use the graphical nature of HTTP links to determine the possible paths through a hypertext system. Although the URL graph-based approaches are effective in prefetching of frequently accessed documents, few of them can prefetch those URLs that are rarely visited. This paper presents a keyword-based semantic prefetching approach to overcome the limitation. It predicts future requests based on semantic preferences of past retrieved Web documents. We apply this technique to Internet news services and implement a client-side personalized prefetching system: NewsAgent. The system exploits semantic preferences by analyzing keywords in URL anchor text of previously accessed documents in different news categories. It employs a neural network model over the keyword set to predict future requests. The system features a self-learning capability and good adaptability to the change of client surfing interest. NewsAgent does not exploit keyword synonymy for conservativeness in prefetching. However, it alleviates the impact of keyword polysemy by taking into account server-provided categorical information in decision-making and, hence, captures more semantic knowledge than term-document literal matching methods. Experimental results from daily browsing of ABC News, CNN, and MSNBC news sites for a period of three months show an achievement of up to 60 percent hit ratio due to prefetching.

Index Terms—NewsAgent, neural networks, personalized news service, prefetching, semantic locality.

1 Introduction

Bandwidth demands for the Internet are growing rapidly with the increasing popularity of Web services. Although high-speed network upgrading has never stopped, clients are experiencing Web access delays more often than ever. There is a great demand for latency tolerance technologies for fast delivery of media-rich Web contents. Such technologies become even more important for clients who connect to the Internet via low-bandwidth modem or wireless links.

Many latency tolerance techniques have been developed over the years. The two most important ones are caching and prefetching. Web traffic characterization studies have shown that Web access patterns from the perspective of a server often exhibit strong temporal locality. By taking advantage of locality, caching, and related content delivery, networks are widely used in server and proxy sides to alleviate Web access latency. In the Web client side, caching is often enabled by default in both popular Internet Explorer and Netscape browsers. However, recent studies showed that benefits from client-side caching were limited due to the lack of sufficient degrees of temporal locality in the Web references of individual clients [19]. The potential for caching requested files was even declining over the years [1], [2]. As a complement to caching, prefetching technique is to prefetch Web documents (more generally, Web objects) that tend to be accessed in the near future while a client is processing previously retrieved documents [21]. Prefetching can also happen offline when the client is not surfing the Web at all as long as the client machine is online. A recent tracing experiment revealed that prefetching could offer more than twice the improvement due to caching alone [19].

Previous studies on Web prefetching were based mostly on the history of client access patterns [6], [12], [23], [28], [29], [30]. If the history information shows an access pattern of URL address A followed by B with a high probability, then B is to be prefetched once A is accessed. The access pattern is often represented by a *URL graph*, where each edge between a pair of vertices represents a follow-up relationship between the two corresponding URLs. The weight on the edge from A to B denotes the probability that Web object B will be requested immediately after object A. To make a prefetching prediction, a search algorithm traverses the graph to find the most heavily weighted edges and paths. The URL graph-based approaches are demonstrated effective in prefetching of Web objects that are often accessed in history. However, there is no way for the approaches to prefetch objects that are newly created or never visited before. For example, all anchored URLs of a page are fresh when a client visits a new Web site and none of them is to be prefetched by any URL graph-based approach. For dynamic HTML sites where a URL has time-variant contents, few existing approaches can help prefetch desired documents.

This paper presents a keyword-based semantic prefetching approach to overcome the limitations and its application in Internet news services. The approach predicts future requests based on semantic preferences of past retrieved

- *C.-Z. Xu is with the Department of Electrical and Computer Engineering, Wayne State University, Detroit, MI 48202. E-mail: czxu@ece.eng.wayne.edu.*
- *T.I. Ibrahim is with FlexBen Corporation, 2250 Butterfield Drive, Suite 100, Troy, MI 48084. E-mail: TIbrahim@flexben.com.*

Manuscript received 16 May 2001; revised 19 June 2002; accepted 21 Feb. 2003.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 114160.

objects, rather than on the temporal relationships between Web objects. It is observed that client surfing is often guided by some keywords in anchor text of Web objects. Anchor text refers to the text that surrounds hyperlink definitions (hrefs) in Web pages. For example, a client with a strong interest in soccer may not want to miss any relevant news with a keyword of "World Cup" in their anchor text. An online shopper with a favorite auto model would like to see all consumer reports or reviews about the model. We refer to this phenomenon as "*semantic locality*." Unlike search engines that take keywords as input and respond with a list of semantically related Web documents, semantic prefetching techniques tend to capture the client surfing interest from his/her past access patterns and predict future preferences from a list of possible objects when a new Web site is visited. Based on semantic preferences, this approach is capable of prefetching articles whose URLs have never been accessed. For example, the approach can help soccer fans prefetch world cup related articles when they enter a new Web site.

Notice that Web HTML format was designed merely for document presentation. A challenge is to automatically extract semantic knowledge of HTML documents and construct adaptive semantic nets between Web documents online. Semantics extraction is a key to Web search engines, as well. Our bitter experience with today's search engines leads us to believe that current semantics extraction technology is far away from maturity for a general-purpose semantic prefetching system. Instead, we demonstrate the idea of semantic prefetching in a special application domain of Internet news services.

We are interested in Internet news services because of the following reasons: First, news headlines are often used as their URL anchors and article titles in most news Web sites. The topic and content of a news article can be captured clearly by a group of keywords in its headline. Second, news service sites like *abcnews.com*, *cnn.com*, and *msnbc.com* often have extensive coverage of the same topics during a certain period. Therefore, articles of interest in different sites can be used to cross-examine the performance of the semantic prefetching approach. That is, one site is used to accumulate URL semantic preferences and the others for evaluation of its prediction accuracy. Finally, effects of prefetching in the news services can be isolated from caching and evaluated separately. Prefetching takes effect in the presence of a cache. Since a client rarely accesses the same news article more than once, there is very little temporal locality in news surfing patterns. This leads to a minimal impact of caching on the overall performance of prefetching. We note that a client who is reading news on a specific topic may need to return to its main page repeatedly for other articles of interest. These repeated visits exhibit certain degrees of temporal locality. However, the caching impact due to this type of locality can be eliminated by the separation of a prefetching cache from the browser's built-in cache. The prefetching cache is dedicated to prefetched documents, while the browser cache exploits the temporal locality between repeated visits of the main pages.

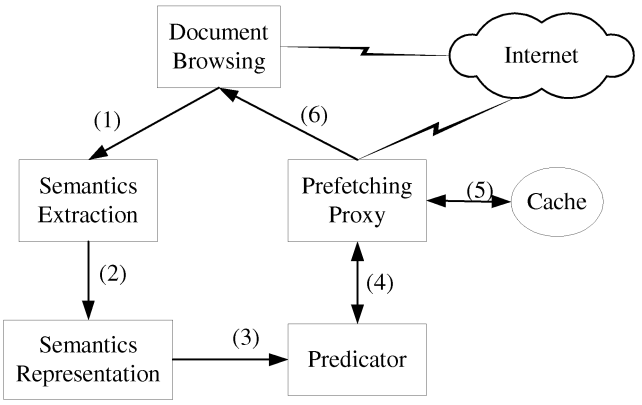


Fig. 1. Architecture of personalized semantic prefetching scheme.

Note that there are two fundamental issues with keyword-based information retrieval: *synonymy* and *polysemy*. Synonymy means that there are many ways of referring to the same concept and polysemy refers to the fact that a keyword may have more than one meaning in different contexts. Latent semantic indexing [10] is often used to address the synonymy issue in a general context of information retrieval. For conservativeness in prefetching, we assume a policy of exact keyword matching in the prediction of client future requests. It is known that all news services organize their articles into categories (e.g., business, sports, and technology). Our semantic prefetching approach takes into account server-provided keyword context (categorical) information to alleviate the impact of keyword polysemy in decision-making and, hence, captures more semantic knowledge than conventional term-document literal matching retrieval methods.

We have implemented the keyword-based semantic prefetching approach in a client-side prefetching prototype system, namely, NewsAgent. It is coupled with a client-side browser, working on behalf of the client. It monitors the client Web surfing activities and establishes keyword-based semantic preferences in different news categories. It employs a neural network model over the keyword set to predict future requests based on the preferences. Such a domain-specific prefetching system is not necessary to be running all the time; it could be turned on or off on-demand like many other advanced features in popular browsers.

The rest of the paper is organized as follows: Section 2 presents an overview of semantic prefetching, followed by a brief description of neural network models for semantics representation. Section 3 describes the NewsAgent architecture and design considerations. Sections 4, 5, and 6 give the details of our experimental methodology and testing results. Related work is discussed in Section 7. Section 8 concludes the paper with remarks on future work.

2 Personalized Semantic Prefetching

2.1 Architecture

Fig. 1 shows the architecture of a generic client-side semantic prefetching scheme. It assumes that a personalized information agent (or proxy) is running behind the Web browser and keeping watch over client surfing

activities. This agent analyzes the documents that its client is browsing with an attempt to understand their semantic knowledge and find out the semantic relationships between the documents (Step 1). The semantic information is represented in an open self-learning capable model that accumulates knowledge about the client preferences (Step 2). Based on the knowledge, the agent makes predictions about client future requests (Step 3) and "steals" network bandwidth to prefetch related Web documents from the Internet (Step 4). The prefetched documents are temporarily stored in the agent's internal cache (Step 5). This prefetching agent also serves as a client-side Web proxy and provides its client with requested Web documents transparently from its internal cache if they are available in the cache (Step 6). Otherwise, it fetches the documents from the Internet. Expectedly, the cache would contain some prefetched documents that are never accessed by the client. This information will be used as feedback to improve prediction accuracy.

This personalized semantic prefetching scheme raises a number of key implementation issues. First is Web document semantics extraction. Web documents are often posted in an HTML presentation format. The problem of extracting semantics of general HTML documents on its own is a challenge. It is fundamental in a broader context of information retrieval on the Internet. Although much effort has been devoted to this issue in both academia and industry over the years, recent studies showed that users remained disappointed with the performance of today's search engines, partly due to certain misunderstanding of document meanings [24].

Automatic semantics extraction for general HTML documents is beyond the scope of this paper. In this paper, we present a domain-specific semantic prefetching approach in Internet news services. Since news headlines are often used as their URL anchors in news services, news article meanings are mostly reflected by the keywords in their URL anchors and their categorical information.

2.2 Neural Network-Based Semantics Model

Semantics extraction aside, another related implementation issue is to develop a semantics representation model that has the capability of self-learning. There are many approaches available for this issue. An example is neural networks. The neural network approach has been around since the late 1950s and come into practical use for general applications since the mid-1980s. Due to their resilience against distortions in the input data and their capability of learning, neural networks are often good at solving problems that do not have algorithmic solutions [15]. In this section, we present neural network basics, with an emphasis on prediction accuracy critical parameters and self-learning rules. Details of its application in semantic prefetching in Internet news services will be shown in Section 3.

2.2.1 Perceptron

A neural network is comprised of a collection of artificial neurons. Each neuron receives a set of inputs and performs a simple function: computing a weighted sum of the inputs and then performing an activation function for decisions.

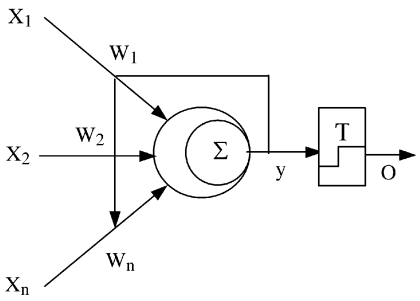


Fig. 2. A graphical representation of a single perceptron.

The activation function can be a threshold operation or a smooth function like hyperbolic operations, depending on the neuron objectives. In order of support predictions in the context of Web document prefetching, we assume a threshold function for the neurons. If the weighted sum of a neuron is less than the predefined threshold, then it yields an output of zero, otherwise one. Specifically, given a set of inputs $x_1, x_2, \ldots x_n$ associated with weights $w_1, w_2, \ldots w_n$, respectively, and a net threshold T, the binary output o of a neuron can be modeled as a hard limiting activation function:

$$o = \begin{cases} 1 & \text{if } y = \sum_{i=1}^{n} w_i \geq x_i T \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

The weighted sum y in (1) is often referred to as *net* of the neuron. Fig. 2 shows a graphical representation of a single perceptron (i.e., neural network with a single neuron). Although it is an oversimplified model of the functions performed by a real biological neuron, it has been shown to be a useful approximation. Neurons can be interconnected together to form powerful networks that are capable of solving problems of various levels of complexity.

In this paper, we assume a single perceptron for a news-prefetching agent. The agent examines the URL anchor texts of a document and identifies a set of keywords for each URL. Using the keywords as inputs, the agent calculates the net output of each URL perceptron. The input $\mathrm{x_i}$ $(1 \leq \mathrm{i} \leq \mathrm{n})$ is set to 1 when its corresponding keyword is present in the anchor text, 0 otherwise. Their nets determine the priority of URLs that are to be prefetched.

2.2.2 Learning Rules

A distinguishing feature of neural networks is self-learning. It is accomplished through learning rules. A learning rule incrementally adjusts the weights of the neuron inputs to improve a predefined performance measure over time. There are two types of learning rules: supervised and unsupervised. Supervised learning algorithms assume the existence of a supervisor who classifies the training patterns into classes, whereas unsupervised learning does not. Supervised learning algorithms utilize the information about class membership of each training pattern as feedback to the perceptron to reduce future pattern misclassification.

In this study, we deploy supervised learning rules because the client of a prefetching agent classifies each input pattern into desired and undesired categories. By comparing its outputs with the client actual requests in the

training phase, the neuron can adjust the input weights accordingly. Usually, the input weights are synthesized gradually and updated at each step of the learning process so that the error between the output and the corresponding desired target is reduced [15].

Many supervised learning rules are available for the training of single perceptron. A most widely used approach is the μ-LMS (least mean square) learning rule. Denote w the vector of the input weights. Denote $\mathrm{d_i}$ the desired (target) output associated with an input pattern and $\mathrm{y_i}$ the actual output of the input pattern. Let

$$J(w) = \frac{1}{2}\sum_{i=1}^{n}(d_i - y_i)^2 \tag{2}$$

be the sum of squared error criterion function. Using the steepest gradient descent to minimize $\mathrm{J(w)}$ gives

$$w^{k+1} = w^k + \mu\sum_{i=1}^{n}(d_i - y_i)x_i, \tag{3}$$

where μ is often referred to as the learning rate of the net because it determines how fast the neuron can react to the new inputs. Its value normally ranges between 0 and 1. The learning rule governed by (3) is called batch LMS rule. The μ-LMS is its incremental version, governed by

$$\begin{cases} w^1 = 0, \\ \mathrm{w^{k+1}} = w^k + \mu(d_i - y_i)x_k. \end{cases}$$

In this study, we assume the prefetching agent deploys the μ-LMS learning rule to update the input weights of its perceptron.

3 NewsAgent: A Personalized News Prefetching Predictor

This section presents details of the keyword-based prefetching technique and architecture of a domain-specific news prefetching predictor: NewsAgent.

3.1 Keywords

A keyword is the most meaningful word within the anchor text of a URL. It is usually a noun referring to some role of an affair or some object of an event. For example, associated with the news "Microsoft is ongoing talks with DOJ" on MSNBC are two keywords "Microsoft" and "DOJ"; the news "Intel speeds up Celeron strategy" contains keywords "Intel" and "Celeron." Note that it is the predictor that identifies the keywords from anchor texts of URLs while users are retrieving documents. Since keywords tend to be nouns and often start with capital letters, these simplify the task of keyword identification.

Recall that there are two fundamental issues with keyword-based information retrieval: synonymy and polysemy. For conservativeness in prefetching, we assume a policy of exact keyword matching in the prediction of client future requests. It is known that the meaning of a keyword is often determined by its context. An interesting keyword in one context may be of no interest at all to the same client if it occurs in other contexts. For example, a Brazil soccer fan may be interested in all the sports news related with the team. In this sports context, the word "Brazil" will guide the fan to all related news. However, the same word may be of less importance to the same client if it occurs in general or world news categories unless he/she has a special tie with Brazil. To deal with this polysemy issue in prefetching, Web documents must be categorized into predefined categories. Automated text categorization is a nontrivial task in any sense. It dates back to the early 1960s and has become an active research topic with the popularity of Web services; a comprehensive survey of state-of-the-art techniques can be found in [31]. We note that the existing categorization algorithms are time costly and supposed to be run at the server side. There are few lightweight algorithms that are suitable for online document classification at the client side. Notice that nearly all news services organize articles into categories (e.g., business, sports, and technology). This server-provided categorical information can be used with keywords to improve the prediction accuracy. A simple strategy is to deploy multiple independent category-specific NewsAgents. Since this categorical information is deterministic, different NewsAgents can be invoked to analyze the URL links from different categories.

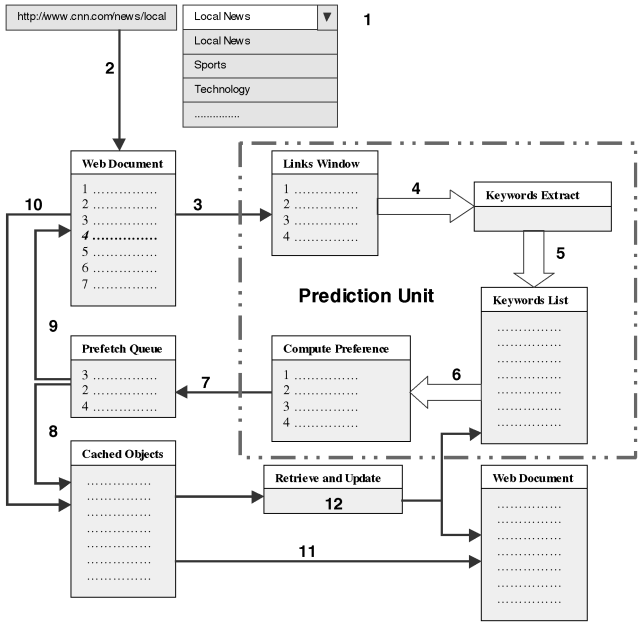


Fig. 3. Architecture of the NewsAgent.

Finally, we note that the significance of a keyword to a client changes with time. For example, news related to "Brazil" hit the headlines during the Brazil economic crisis period in 1998. The keyword should be sensitive to any readers of market or financial news during the period. After the crisis was over, its significance decreased unless the reader has special interests in the Brazilian market. Due to the self-learning capability of a neural network, the keyword weights can be adjusted to reflect the change.

3.2 NewsAgent Architecture

Fig. 3 shows the architecture of NewsAgent. Its kernel is a prediction unit. It monitors the way of its client browsing activities. For each required document, the predictor examines the keywords in its anchor text, calculates their preferences, and prefetches those URL links that contain keywords with higher preferences than the perceptron

threshold. Specifically, the system has the following the workflow:

1. User requests a URL associated with a category.
2. The URL document is retrieved and displayed to the user.
3. The contents of the document are examined to identify the first set of links to evaluate.
4. Each link from the set is processed to extract the keywords.
5. Keywords are matched to the keyword list. They are added to the list if they occur for the first time.
6. The corresponding preferences are computed using the weights associated with the keywords.
7. Links with preferences (i.e., nets) higher than the neuron threshold are sorted and placed in the prefetching queue.
8. Links are accessed one by one and the objects are placed in the cache.
9. If the prefetching queue is empty, process the next set of links (Step 3).
10. When the user requests a link, the cache list is examined to see if it's there.
11. If it is there, then it is displayed.
12. If not, it is retrieved from the Web and the corresponding keyword's weights are updated.

In most cases, a Web page contains a large number of hyperlinks. The NewsAgent examines the URL links by groups. We assume each group contains a fixed number of links and refer to the group size as *prefetching breadth*. Evaluation of the embedded links by groups is patterned after client browsing behaviors because a client tends to look at a few hyperlinks before he/she decides which to follow. Recall that prefetching needs to be carried out during the time when the client is viewing a page. A recent analysis of client access patterns showed that the average rate that http requests were made was about two requests per minute [2]. In a very short time, the NewsAgent may not be able to complete the evaluation of all embedded links and the prefetch of those with high preferences in the current page. The group evaluation strategy provides a quick response to client Web surfing and possibly increases the hit ratio due to prefetching. For example, consider a page containing a large number of URLs. Assume the last URLs in the list have nets of more than 90 percent and a URL in the beginning has a net of more than 80 percent. The NewsAgent based on the group evaluation strategy will prefetch the moderately preferred link first to meet its client's possible immediate needs. The highest preferred links will be prefetched as the client browsing moves forward.

Once a request for an anchored URL is received, the NewsAgent starts to look for the requested pages in the cache. If it has already been available due to prefetching, the page is provided for display. (The weights of its related keywords will remain unchanged by the rule of (4).) Otherwise, the NewsAgent suspends any ongoing prefetching processes and starts to examine the anchor texts of the requested link for keywords. If a keyword is introduced for the first time, a new weight will be assigned to it. Otherwise, its existing weight will be *positively* updated $(d = 1)$ by the rule of (4).

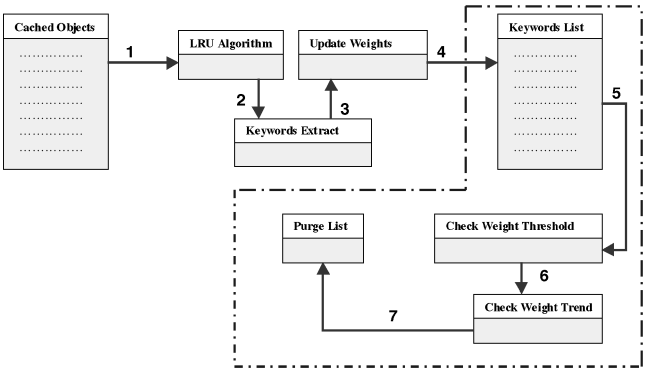


Fig. 4. Overflow control for the keyword list and prefetching cache.

3.3 Control of Prefetching Cache and Keyword List

Notice that the keyword list could be full with unwanted items when the client loses interest in a long-standing topic. The NewsAgent internal cache may also contain undesired documents due to the presence of some trivial (nonessential) keywords in the anchor texts. For example, the keyword "DOJ" is trivial in a sector of technology news. It is of interest only when it appears together with some other technology-related keywords like Microsoft or IBM. Fig. 4 shows a mechanism of the NewsAgent to control the overflow of the keyword list and the document prefetching cache:

1. When the cache list is full, use the LRU algorithm to remove the least recently accessed objects.
2. Extract the corresponding keywords.
3. Negatively update those weights.
4. Compare the updated weights to the minimum weight threshold.
5. If they are larger than the minimum, check the updating trend (decreasing in this case).
6. Purge those keywords from the list if they are below the minimum weight threshold.
7. If the keyword list is full, consider Steps 5, 6, and 7.

However, the trend will not be necessarily decreasing in this case.

By checking the cache, we can identify those prefetched documents that have never been accessed. The NewsAgent will then decrement the weights of their related keywords. Also, we set a *purging threshold* to determine whether a keyword should be eliminated from the keyword list or not. If the weight of a keyword is less than this threshold, we examine its change *trend*. The weight change trend is a flag that indicates the keyword weight is increasing or decreasing. If it is negative, then the keyword is considered trivial and is eliminated from the list.

The keywords are purged once the keyword list is full. The list size should be set appropriately. A too small list will force the prefetching unit to purge legitimate keywords, while a too large list would be filled with trivial keywords, wasting space and time in evaluation. We set the list size to 100 and the purging threshold value to 0.02 in our experiments. The purging threshold was set based on our preliminary testing, which will be discussed in Section 6.3. These settings are by no means optimal choices.

Prefetched documents need to be replaced when the cache has no space to put new documents. The documents

to be purged are those that have been in the cache for a long time without being accessed. The URL anchor texts (i.e., news captions) that led to their prefetching are first examined. The weights of their related keywords are then *negatively* adjusted $(d = 0)$ by the rule of (4). This assures that these keywords would unlikely trigger the prefetching unit again in the near future.

4 Experimental Methodology

The NewsAgent system was developed entirely in Java and integrated with a pure Java Web browser, IceBrowser from www.IceSoft.com. This lightweight browser with its open source code gave us a great flexibility in the evaluation of the NewsAgent system. Using such a lightweight browser simplified the process of setting experiment parameters because the browser just did what it was programmed for and there were no hidden functions or Add-Ins to worry about, as those with full-blown browsers like Netscape and Explorer. Although the NewsAgent prototype was integrated with the IceBrowser, it maintained its own cache so as to dedicate this cache to prefetched documents.

Since the Web is essentially a dynamic environment, its varying network traffic and server-load make the design of a repeatable experiment challenging. There were studies dedicated to characterizing user access behaviors [2] and prefetching/caching evaluation methodologies; see [9] for a survey of these evaluation methods. Roughly, two major evaluation approaches are available: simulation based on user access traces and real-time simultaneous evaluation. A trace-based simulation works like an instruction execution trace-based machine simulator. It assumes a parameterized test environment with varying network traffic and workload. It is a common approach to receive repeatable results. There are benchmarks about client access patterns [2]. However, all the benchmarks that are available to the public are about aggregate behaviors of client groups. Due to privacy concerns, there won't be any open traces about individual clients. Needless to say, client-side benchmarks with specific URL contents. Moreover, current implementation of the NewsAgent is domain specific and it should be turned on only when its client gets to read news. To the best of our knowledge, there are no such domain-specific traces available either.

The real-time simultaneous evaluation approach [8] was initially designed for the evaluation of Web proxy strategies. In this approach, the proxy under test is running in parallel with another reference proxy. Both proxies perform the same sequence of accesses simultaneously so that more time-variant test environment factors like access latency can be considered. Along the lines of this approach, we evaluated the NewsAgent performance by accessing multiple news sites simultaneously. One news site was accessed from a prefetching-enabled browser to train the NewsAgent. That is, the NewsAgent kept accumulating knowledge about its client's access preferences when he/she was reading news from the site. The trained NewsAgent then went to work with another browser that has no built-in prefetching functions to retrieve news items from other sites.

All the experimental results were collected by one of the authors while he was reading news from major online news services in different periods. Since this prefetching approach relies upon the reading interest of individual clients accumulated in training phases, results from different clients are incomparable. The test environment of this paper was a Pentium II 600 MHz Windows NT machine with 128 MB of RAM connected to the Internet through a T-1 connection. Following are the experimental settings:

- Topic selection: It is expected that the performance of semantics-based prefetching systems be determined by semantic locality of the access patterns. In order to establish the access patterns systematically, we selected topics of interest to guide the client's daily browsing behaviors. We are interested in those hot topics that have extensive coverage in major news sites for a certain period so that we can complete the experimentation before the media changes its interest. We conducted two experiments in two different periods when the Kosovo crisis and the Clinton affair were at their peak. Both topics lasted long enough in the media for prefetching training and subsequent evaluation. Notice that the NewsAgent has no intrinsic requirements for the selection of topics in advance. The client interest is expected to change over time and the algorithm works in the same way no matter whether there is a selected topic or not. However, without a preselected topic in evaluation, most of the keywords generated from a random browsing would be too trivial to be purged. As a result, there would be few prefetching activities over a long period due to the lack of semantic locality.
- Site selection: We selected cnn.com for the NewsAgent training and abcnews.com and msnbc.com for its evaluation. Each phase took a month (1 Apr. 1999 to 5 May 1999 for training and 7 May 1999 to 14 June 1999 for evaluation).
- Client access pattern: The client access pattern was monitored and logged daily. Also, the time between two consecutive access requests (i.e., the time allowed to view a page) was set to an average of one minute according to a recent study of client-side Web access patterns [2].

5 Experimental Results

As indicated earlier in the previous section, the experiment was conducted in two phases: training and evaluation. This section presents the results of each phase.

5.1 Training Phase

The NewsAgent system relies on a single perceptron to predict its client future request. It follows the μ-LMS learning rule to adapt to the client interest change. Before its deployment, the NewsAgent must be trained. During its training phase, the NewsAgent monitors the client accesses to cnn.com server, analyzes the related URL anchor texts, and identifies embedded keywords as inputs to its neural network learning part. It predicts the client access preferences and compares its predictions with the client actual access behaviors. Using the comparative results as a feedback, the NewsAgent keeps adjusting the keyword weights recursively. In general, a large set of training data

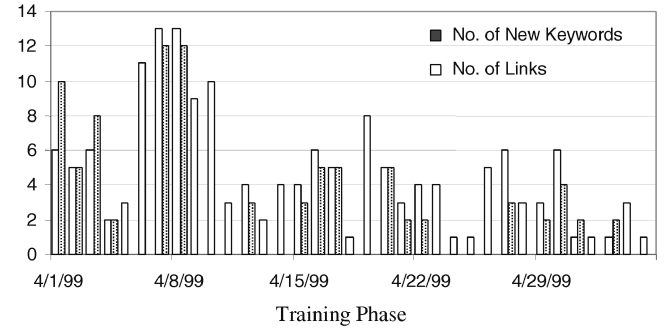


Fig. 5. Access patterns in the training session.

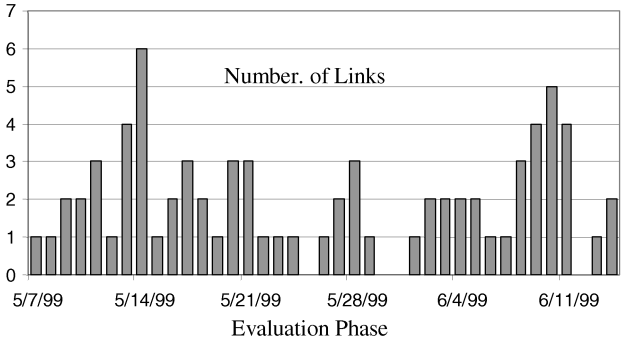


Fig. 7. Access patterns in the evaluation session.

for neural networks with a self-learning capability would lead to highly accurate predictions. However, this may not be the case in news prefetching training. It is because the topics interesting to the client change with time and coverage of a new subject would introduce a set of new keywords. It is the dynamic nature of keyword inputs that makes the NewsAgent adaptive to the change of client interests.

During this training phase from 1 Apr. 1999 to 5 May 1999, a total of 163 links were accessed. The average link size was 18 kilobytes with a minimum of 9.4 kilobytes and a maximum of 33.4 kilobytes. Fig. 5 shows the daily access patterns during this period.

From Fig. 5, it can be seen that in the first four days (1 Apr. 1999 to 4 Apr. 1999), the NewsAgent identified many new keywords as more links were accessed. This is expected because the NewsAgent started with an initialized neuron with no keyword inputs. The next two days (5 Apr. 199 tp 6 Apr. 1999) found no new keywords, although the media had more coverage about the topic we selected. As time proceeds, the number of new keywords increased sharply in the next two days (7 Apr. 1999 to 8 Apr. 1999) because of new related breaking news. From the figure, it can be seen that the rise-and-drop pattern occurs repeatedly and demonstrates an important self-similarity characteristic. Due to the dynamics nature of news subjects, whenever a client (or the media) loses interests in a topic, emergence of new host topics would introduce another set of new keywords and the self-similarity pattern repeats.

In the NewsAgent training phase, we set its prediction threshold rate T, as defined in (1), to a typical median value 0.5. That is, a neuron output of 0.5 or higher was considered a success matching between the neuron prediction and client desire. We also set the NewsAgent learning rate, as defined in (4), to 0.1. This learning rate of 0.1 tends to maintain the stability of the neuron and guarantee a smooth weight updating process. In the next section, we will show that high learning rate values can lead to instability and, subsequently, the failure of unit. Fig. 6 plots the success rate of the neuron against the number of links accessed. As we can see, the success rate increases with the number of links accessed until it saturates around 60 percent. This implies that, after a certain training time, the unit could identify 6 out of 10 links as links of interest. Since topic changes tend to generate additional sets of keywords to the neuron, we would not expect an extremely high success rate.

5.2 Evaluation Phase

The NewsAgent was trained during its client was accessing to cnn.com news site. After having been trained, it entered into its second evaluation phase in the access of abcnews.com new site. During the evaluation phase, the NewsAgent kept its keyword inputs and their weights unchanged. Since both cnn.com and abcnews.com were covering similar news topics, we expected that the trained NewsAgent would prefetch the client's preferred Web documents from abcnews.com according to the knowledge accumulated from the training with cnn.com. Fig. 7 shows the access pattern in this phase.

Prefetching systems are often evaluated in terms of four major performance metrics: *hit ratio*, *byte-hit ratio*, *waste ratio*, and *waste byte ratio*. Hit ratio refers to the percentage of requested objects that are found in the prefetching cache. Byte-hit ratio refers to the percentage of hit links in size. This pair of metrics, also referred to as recall, measures the usefulness of prefetching in terns of Web access latency improvement. Waste ratio refers to the percentage of undesired documents that are prefetched into the cache.

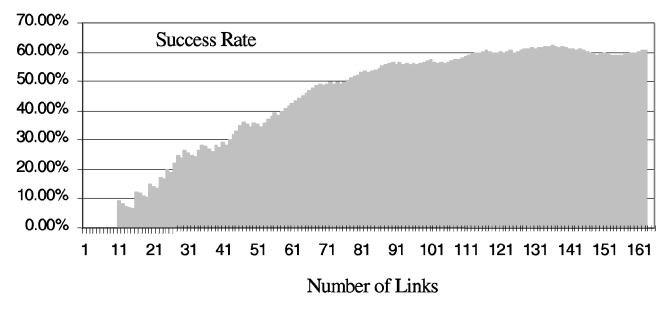


Fig. 6. Change of the neuron success rate with more links being accessed.

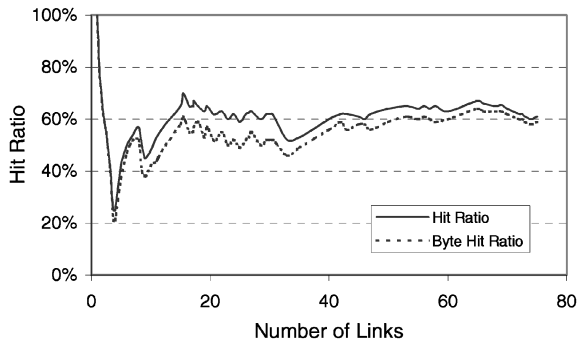


Fig. 8. Hit ratio and byte hit ratio in the evaluation session.

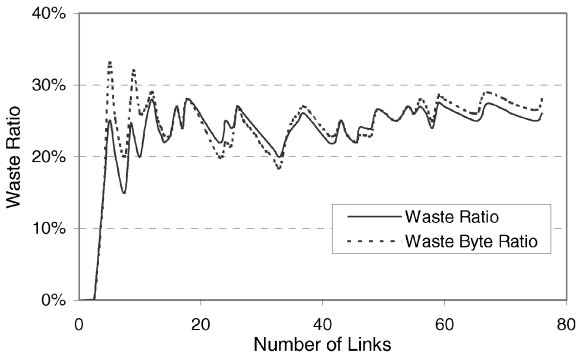


Fig. 9. Waste ratio and byte waste ratio in the evaluation session.

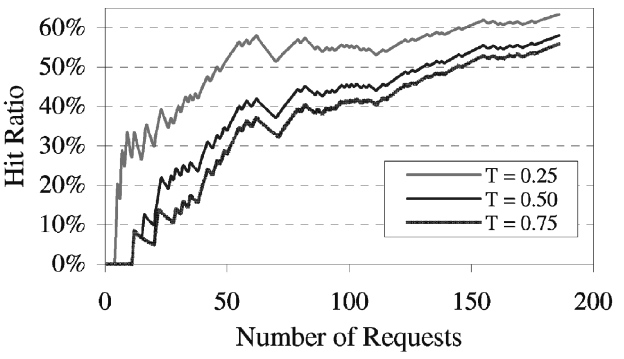


Fig. 10. Effects of the threshold on cache hit ratio.

Byte waste ratio refers to the percentage of undesired documents in size. This pair of metrics reflects the cost (i.e., the extra bandwidth consumption) for prefetching.

Fig. 8 shows the hit and byte hit ratios in this phase as a function of number of accessed links. While the news in abc.com had never been accessed before by the client, the NewsAgent achieved a hit rate of around 60 percent based on the semantic knowledge accumulated in the visit to cnn.com. Since the news pages have a small difference in size, the plot of the byte hit ratio matches with that of the hit ratio. Since the topic of Kosovo we selected lasted longer than our experiment period, the keywords that were identified in the training phase remained the leading keywords in the evaluation phase. Since the input keywords and their weights in the NewsAgent neuron remained unchanged in the evaluation phase, it is expected that accuracy of the prediction will decrease slowly as the client being distracted from the selected topic.

Fig. 9 shows an average of 24 percent waste ratio and approximately the same percentages of byte waste ratio during the evaluation phase. This is because many of the links contained similar material and the reader might be satisfied with only one of them. Also, in the evaluation phase, the process of updating weights and purging trivial keywords was not active. However, the neuron kept prefetching those undesired documents as long as the output of the neuron was greater than the net threshold. This leads to a higher waste ratio as well.

6 Effects of Net Threshold and Learning Rate

Accuracy of the predictive decisions is affected by a number of parameters in the NewsAgent. To evaluate the effects of the parameters, particularly net threshold and learning rate on hit ratio and waste ratios, we implemented multiple predictors with identical parameter settings, except the one under test, with the same inputs and access sequence. We conducted the experiment over two new sites: *Today in America* at msnbc.com and the *US News* at cnn.com/US during the period from 18 Nov. 1998 to 16 Dec. 1998, five days a week, and two half-an-hour sessions a day. The first contained multimedia documents, while the other was a combination of plain texts and image illustrations. Results from the surfing of these two news sites not only revealed the impact of the parameters, but also helped verify the observations we obtained in the last experiment.

During the experiment time, the topic of "Clinton affair" had much coverage in both sites. As we explained in the preceding section, we selected a guiding hot topic in evaluation because its extensive coverage would help collect more data within a short time period. The NewsAgent can identify keywords and establish client access patterns automatically by monitoring his/her browsing behaviors. To reflect the fact that clients may be distracted from their main topic of interest for a while in browsing, we intentionally introduced noise by occasionally accessing headlines that did not belong to the selected topic. We accessed a random link every 10 trials. We also assumed that each access will be followed by an average of one minute viewing, according to a recent analysis of client-side Web access patterns [2].

6.1 Effect of Net Threshold

Fig. 10 plots the hit ratios due to prefetching with neuron thresholds of 0.25, 0.50, and 0.75, respectively. The figure shows that a low threshold would yield a high hit ratio for a large cache. This agrees with what (1) reveals. Since an extremely small threshold could trigger the prefetching unit frequently, it would lead to prefetch of most of the objects embedded in a given Web page. By contrast, the higher the threshold, the smoother the curve is and the more accurate the prediction will be.

Fig. 11 shows the waste ratio in the cache. Expectedly, the lower the threshold value, the more undesired objects that get cached. The cache may be filled up with undesired objects quickly. There is a trade off between the hit ratio and utilization ratio of the cache. The optimum net threshold should strike a good balance between the two objectives.

6.2 Effects of Learning Rate

From the learning rule of (4), it is known that the learning rate controls the adaptivity of the NewsAgent to keyword inputs. Fig. 12 shows the hit ratios due to different learning rates. From the figure, it can be seen that a high learning

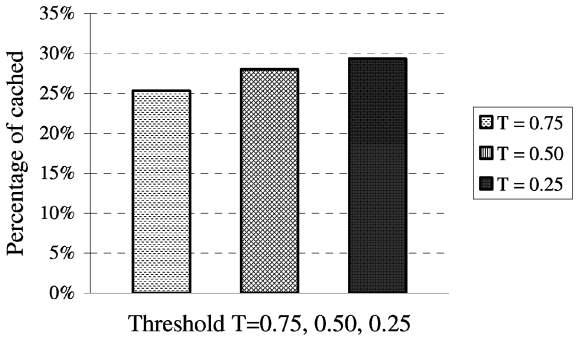


Fig. 11. Effects of the threshold on cache waste ratio.

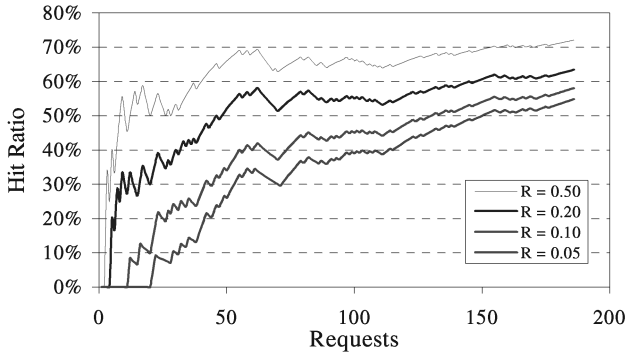


Fig. 12. Effect of the learning rate on cache hit ratio.

rate leads to a high hit ratio. It is because high learning rates tend to update the weights quickly. Ripples on the curve are due to the fast updating mechanism.

The figure also shows that the smaller the learning rate, the smoother the curve and the more accurate the prediction will be. It is because the weights of trivial keywords are updated with minimal magnitude. This generally affects cache utilization because insignificant weights won't trigger the prediction unit with a lower learning rate. This may not be the case for a unit with a higher learning rate, as shown in Fig. 13. Thus, choosing a proper learning rate depends mainly on how fast we want the prediction unit to respond and what percentage of the cache waste we can tolerate.

6.3 Update of Keyword and Purge of Trivial Keywords

Recall that the prefetching unit relies on a purging mechanism to identify trivial weights and decrease their impact on the prediction unit so as not to be triggered by accident. In the experiments, we found that, on average, approximately four to five trivial keywords were introduced for each essential keyword. Fig. 14 plots the distribution of the keyword weights associated with the NewsAgent. The figure shows the system kept the trivial weight minimized as the browsing process continues.

As outlined in Fig. 4, the keyword weights are reduced whenever their related objects in the cache are to be replaced by the prefetching unit. The objects that have never been accessed have the lowest preference value and are to be replaced first. Accordingly, weights of the keywords associated with the replaced object were updated. This guarantees that trivial keywords had minimum impacts on prefetching decisions.

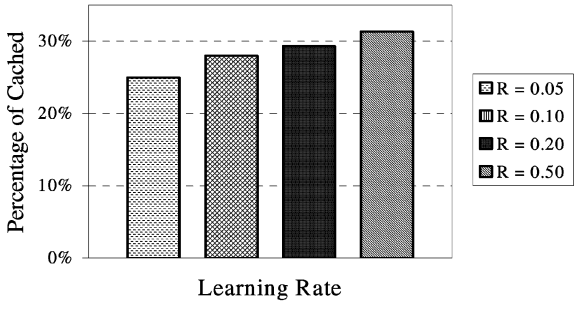


Fig. 13. Effect of the learning rate on cache waste ratio.

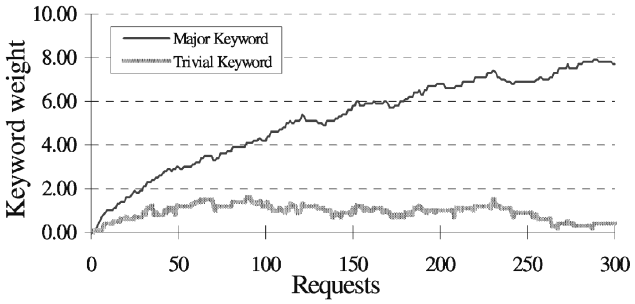


Fig. 14. Effect of unrelated keyowrds on prediction accuracy.

7 Related Work

The concept of prefetching has long been used in a variety of distributed and parallel systems to hide communication latency [5]. A closely related topic to prefetching in Web surfing is file prefetching and caching in the design of remote file systems. Shriver and Small presented an overview of file system prefetching and explained with an analytical model why file prefetching works [32]. A most notable work in automatic prefetching is due to Griffioen and Appleton [14]. They proposed predicting future file accesses based on past file activities that are characterized by probability graphs. Padmanabhan and Mogul adapted the Griffioen-Appleton file predictive algorithm to reduce Web latency [28].

Web prefetching is an active topic in both academia and industry. Algorithms used in today's commercial products are either greedy or user guided. A greedy prefetching technique does not keep track of any user Web access history. It prefetches documents based solely on the static URL relationships in a hypertext document. For example, WebMirror [25] retrieves all anchored URLs of a document for a fast mirroring of the whole Web site; NetAccelerator [17] prefetches the URLs on a link's downward path for a quick copy of HTML formatted documents. The user-guided prefetching technique is based on user-provided preference URL links. There were a number of industry examples in this category, but few succeeded in marketing.

Research in academia is mostly targeted at intelligent predictors based on user access patterns. The predictors can be associated with servers, clients, or proxies. Padmanabhan and Mogul [28] presented a first server-side prefetching approach based on the Griffioen and Appleton algorithm. In this approach, client access patterns are depicted as a weighted *URL dependency graph*, where each edge represents a follow-up relationship between a pair of URLs and the edge weight denotes the transfer probability from one URL to the other. The server updates the dependency graph dynamically as requests come from clients and makes predictions of future requests. Schechter et al. [30] presented methods of building a sequence prefix tree (*path profile)* based on HTTP requests in server logs and using the longest matched most-frequent sequence predict next requests. Sarukkai proposed a probabilistic sequence generation model based on Markov chains [29]. On receiving a client request, the server uses the HTTP probabilistic link prediction to calculate the probabilities of the next requests based on the history of requests from the same client.

Loon and Bharghavan proposed an integrated approach for proxy servers to perform prefetching, image filtering,

and hoarding for a mobile client [23]. They used usage profiles to characterize user access patterns. A usage profile is similar to the URL dependency graph, except that each node is associated with an extra weight indicating the frequency of access of the corresponding URL. They presented heuristic weighting ways to reflect the changing access patterns of users and temporal locality of accesses in the update of usage profiles. Markatos and Chronaki [26] proposed a top-10 prefetching technique for servers to push their most popular documents to proxies regularly according to clients' aggregated access profiles. A question is that high hit ratio in proxies due to prefetching may give servers a wrong perception of future popular documents.

Fan et al. [12] proposed a proxy-initiated prefetching technique between a proxy and clients. It is based on a Prediction-by-Partial-Matching (PPM) algorithm originally developed in data compressor [7]. PPM is a parametric algorithm that predicts the next l requests based on past m accesses by the user and limits the candidates by a probability threshold of t. PPM is reduced to the algorithm of Papadumanta and Mogul when m is set to 1. Fan et al. showed that the PPM prefetching algorithm performs best when $m = 2$ and $l = 4$. A similar idea was recently applied to prefetching of multimedia documents as well by Su et al. [33].

From the viewpoint of clients, prefetching is used to speed up Web access, particularly to those objects that are expected to experience long delays. Klemm [18] suggested prefetching Web documents based on their estimated round-trip retrieval time. Since their technique assumes a priori knowledge about the sizes of prospective documents for estimation of their round-trip time, it was limited to prefetching of static Web pages.

Prefetching is essentially a speculative process and incurs a high cost when its prediction is wrong. Cunha and Jaccoud [6] focused on the decision of when prefetching should be enabled or not. They constructed two user models based on a random walk approximation and digital signal processing techniques to characterize user behaviors and help predict users' next actions. They proposed coupling the models with prefetching modules for customized prefetching services. Crovella and Barford [4] studied the effects of prefetching on network performance. They showed that prefetching might lead to an increase of traffic burstiness. They proposed using transport rate-controlled mechanisms to reduce smooth traffic caused by prefetching.

Previous studies on intelligent prefetching were based mostly on temporal relationships between accesses to Web objects. Semantics-based prefetching was studied in the context of file prefetching. Kuenning and Popek suggested using semantic knowledge about the relationship between different files to hoard data onto a portable computer before the machine is disconnected [20]. Lei and Duchamp proposed capturing intrinsic corrections between file accesses in tree structures online and then match against existing pattern trees to predict future access [22]. By contrast, we deployed neural nets over keywords to capture the semantic relationships between Web objects and improve prediction accuracy by adjusting the weights of keywords online.

8 Concluding Remarks

This paper has presented a keyword-based semantic prefetching approach in the context of Internet news services to tolerate Web access latency from the perspective of clients. It exploits semantic preferences of client requests by analyzing keywords in the URL anchor texts of previously accessed articles in different news categories. It employs a neural network model over the keyword set to predict future requests. Unlike previous URL graph-based techniques, this approach takes advantage of the "semantics locality" in client access patterns and is capable of prefetching documents that have never been accessed before. It also features a self-learning capability and good adaptability to the change of client surfing interest. Based on the approach, a client-side personalized prefetching prototype, NewsAgent, was implemented. The system was evaluated in daily browsing of three major news sites over a period of three months. The experimental results showed an achievement of up to 60 percent hit ratio due to the prefetching approach. This is close to the impact of general Web caching, although news surfing activities benefit a little from client-side caching.

Future work would focus on the following aspects. First is to construct a prediction unit using neural networks, instead of single perceptron. Besides the keywords, other factors, such as document size and the URL graphs of a hypertext document, will also be taken into account in the design of the predictor. Second is to improve cache utilization. We notice that the hit ratio saturates at around 60-70 percent after a reasonable number of requests. More studies need to be conducted in this aspect to reduce the cache waste ratio. On a different track altogether, other statistical classification and learning techniques like Bayesian networks could be deployed.

Like many domain-specific search engines or focused crawling [3], [13], [27], the NewsAgent system is targeted at an application domain of Internet news services. However, the idea of semantic prefetching is applicable to other application domains, such as online shopping and auction, where semantics of Web pages are often explicitly presented in their URL anchor texts. Such domain-specific prefetching facilities can be turned on or off on-demand.

The NewsAgent system is based on keyword matching. Two issues related to keywords are synonymy and polysemy. As we argued in Section 3.1, we do not exploit keyword synonymy for conservativeness in prefetching. We alleviate the impact of polysemy by taking advantage of the categorical information in news services. In a broader context of information retrieval, there exist many techniques to address these two issues; see [24] for a comprehensive review. Most notable is latent semantic indexing (LSI) [10]. It takes advantage of the implicit higher-order structure of the association of terms with articles to create a multidimensional semantic structure of the information. Through the pattern of co-occurrences of words, LSI is able to infer the structure of relationships between article and words. Due to its high-time complexity in inference, its application to real-time Web prefetching requires a trade off between its time cost and prediction accuracy.

Recently, XML and RDF (Resource Description Framework) are emerging as standards of data exchange format in support of semantic interoperability on the Web [11]. With the popularity of XML or RDF compliant Web resources, we expect semantics-based prefetching techniques will play an increasingly important role in reducing Web access latency and will lead to more personalized quality services on the semantic Web.

ACKNOWLEDGEMENTS

The authors would like to thank the editor, Dr. Louiqa Raschid, and anonymous referees of this paper for their thoughtful reviews and constructive comments. They would also like to thank Dr. Bill Grosky and Dr. Guihai Chen for their helpful discussions on early versions of this work. Part of the early results were presented in the *Proceedings of the 20th IEEE Conference on Distributed Computing Systems*, April 2000 [16]. This material is based up work supported in part by the US National Science Foundation under Grants CCR-9988266 and ACI-0203592. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the US National Science Foundation.

REFERENCES

[1] V. Almeida, A. Bestavros, M. Crovella, and A. de Oliveira, "Characterizing Reference Locality in the WWW," *Proc. IEEE Conf. Parallel and Distributed Information Systems,* pp. 92-103, Dec. 1996.

[2] P. Barford, A. Bestavros, A. Bradley, and M. Crovella, "Changes in Web Client Access Patterns: Characteristics and Caching Implications," *World Wide Web: Special Issue on Characterization and Performance Evaluation,* vol. 2, pp. 15-28, 1999.

[3] S. Chakrabarti, M. van der Berg, and B. Dom, "Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery," *Proc. Eighth Int'l World Wide Web Conf. (WWW8),* pp. 545-562, May 1999.

[4] M. Crovella and P. Barford, "The Network Effects of Prefetching," *Proc. IEEE Infocom'98,* pp. 1232-1240, Mar. 1998.

[5] D. Culler, J. Singh, and A. Gupta, *Parallel Computer Architecture: A Hardware/Software Interface.* Morgan Kaufmann, 1998.

[6] C.R. Cunha and C.F.B. Jaccoud, "Determining WWW User's Next Access and Its Application to Prefetching," *Proc. Int'l Symp. Computers and Comm.,* pp. 6-11, July 1997.

[7] K.M. Curewitz, P. Krishnan, and J. Vitter, "Practical Prefetching via Data Compression," *Proc. SIGMOD '93,* pp. 257-266, May 1993.

[8] B. Davison, "Simultaneous Proxy Evaluation," *Proc. Fourth Int'l Web Caching Workshop,* pp. 170-178, Mar. 1999.

[9] B. Davison, "A Survey of Proxy Cache Evaluation Techniques," *Proc. Fourth Int'l Web Caching Workshop,* pp. 67-77, Mar. 1999.

[10] S. Deerwester et al. "Indexing by Latent Semantic Analysis," *J. Am. Soc. Information Science,* vol. 41, no. 6, pp. 391-407, 1990.

[11] S. Decker, S. Melnik et al. "The Semantic Web: The Role of XML and RDF," *IEEE Internet Computing,* pp. 63-74, Sept./Oct. 2000.

[12] L. Fan, P. Cao, W. Lin, and Q. Jacobson, "Web Prefetching between Low-Bandwidth Clients and Proxies: Potential and Performance," *Proc. SIGMETRICS '99,* pp. 178-187, May 1999.

[13] E.J. Glover, G.W. Falke, S. Lawrence, W. Birmingham, A. Kruger, C. Giles, and D. Pennock, "Improving Category Specific Web Search by Learning Query Modifications," *Proc. Symp. Applications and the Internet (SAINT 2001),* Jan. 2001.

[14] J. Griffioen and R. Appleton, "Reducing File System Latency Using a Predictive Approach," *Proc. 1994 Summer USENIX Conf.,* pp. 197-207, June 1994.

[15] H. Hassoun, *Fundamentals of Artificial Neural Networks.* The MIT Press, 1995.

[16] T. Ibrahim and C. Xu, "Neural Network Based Prefetching to Tolerate WWW Latency," *Proc. 20th IEEE Conf. Distributed Computing Systems,* pp. 636-643, Apr. 2000.

[17] ImsiSoft, NetAccelerator Product Overview, available at http://www.netaccelerator.net, 2003.

[18] R.P. Klemn, "WebCompanion: A Friendly Client-Side Web Prefetching Agent," *IEEE Trans. Knowledge and Data Eng.,* vol. 11, no. 4, pp. 577-594, July/Aug. 1999.

[19] T. Kroeger, D. Long, and J. Mogul, "Exploring the Bounds of Web Latency Reduction from Caching and Prefetching," *Proc. USENIX Symp. Internet Technologies and Systems,* pp. 13-22, Dec. 1997.

[20] G. Kuenning and G. Popek, "Automated Hoarding for Mobile Computers," *Proc. ACM Symp. Operating Systems Principles,* pp. 264-275, Oct. 1997.

[21] D. Lee, "Methods for Web Bandwidth and Response Time Improvement," *World Wide Web: Beyond the Basics,* M. Abrams, ed., chapter 25, Prentice Hall, Apr. 1998.

[22] H. Lei and D. Duchamp, "An Analytical Approach to File Prefetching," *Proc. USENIX 1997 Ann. Technical Conf.,* pp. 275-288, Jan. 1997.

[23] T.S. Loon and V. Bharghavan, "Alleviating the Latency and Bandwidth Problems in WWW Browsing," *Proc. USENIX Symp. Internet Technologies and Systems,* pp. 219-230, Dec. 1997.

[24] M. Kobayashi and K. Takeda, "Information Retrieval on the Web," *ACM Computing Surveys,* vol. 32, no. 2, pp. 144-173, June 2000.

[25] Macca Soft, WebMirror Product Overview, available at http://www.maccasoft.com, 2002.

[26] E.P. Markatos and C.E. Chronaki, "A Top-10 Approach to Prefetching on the Web," *Proc. INET '98,* July 1998.

[27] A. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Building Domain-Specific Search Engines with Machine Learning Techniques," *Proc. AAAI Spring Symp. Intelligent Agents in Cyberspace,* 1999.

[28] V. Padmanabhan and J. Mogul, "Using Predictive Prefetching to Improve World Wide Web Latency," *Computer Comm. Rev.,* vol. 26, no. 3, pp. 22-36, July 1996.

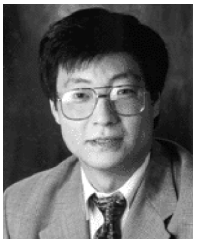
[29] R. Sarukkai, "Link Prediction and Path Analysis Using Markov Chains," *Proc. Ninth Int'l World Wide Web Conf.,* 2000.

[30] S. Schechter, M. Krishnan, and M. Smith, "Using Path Profiles to Predict HTTP Requests," *Proc. Seventh Int'l World Wide Web Conf.,* also appeared in *Computer Networks and ISDN Systems,* vol. 20, pp. 457-467, 1998.

[31] F. Sebastiani, "Machine Learning in Automated Text Categorization," *ACM Computing Surveys,* vol. 34, no. 1, pp. 1-47, Mar. 2002.

[32] E. Shriver and C. Small, "Why Does File System Prefetching Work?" *Proc. 1999 USENIX Ann. Technical Conf.,* June 1999.

[33] Z. Su, Q. Yang, and H. Zhang, "A Prediction System for MultiMedia Prefetching in Internet," *Proc. ACM Multimedia,* 2000.



Cheng-Zhong Xu received the BS and MS degrees from Nanjing University, China, in 1986 and 1989, respectively, and the PhD degree from the University of Hong Kong, Hong Kong, China, in 1993, all in computer science. He is currently an associate professor in the Department of Electrical and Computer Engineer at Wayne State University. Dr. Xu has published more than 60 peer-reviewed papers in various journals and conference proceedings in the areas of resource management in distributed and parallel systems, high-performance cluster computing, mobile agent technology, and Web semantics. He is the leading coauthor of the book *Load Balancing in Parallel Computers: Theory and Practice* (Kluwer Academic, 1997). He has served on technical program committees of numerous international conferences, including IEEE HiPC 2002, IEEE ICDCS 2001, ICDCS 2004, and ICPP 2004. He was a guest coeditor for a special issue on scalable Internet services and architecture for the *Journal of Parallel and Distributed Computing* in 2003. He is a recipient of the Faculty Research Award of Wayne State University in 2000, the President Award for Excellence in Teaching in 2002, and the Career Development Chair Award in 2003. He is a senior member of the IEEE.

Tamer I. Ibrahim received the BS degree from the United Arab Emirates University, United Arab Emirates in 1996, at the graduating top of his class in the Department of Electrical Engineering, and the MS degree in 2000 from Wayne State University in the Department of Electrical and Computer Engineering. His main research interests are primarily in the fields of parallel and distributed systems with applications to Internet technologies. From 1996 to 1997, he worked as a teaching assistant in the Department of Electrical Engineering at the UAE University. In 1998, he joined a Michigan-based private company as an IT consultant where he is currently the chief software engineer in the IT department. His responsibilities include the design and the implementation of Internet/Intranet-based business applications and systems integration.