International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 2– No.1, May 2012 – www.ijais.org

Adaptive Web Prefetching Scheme using Link Anchor Information

P. Venketesh
Assistant Professor (SG)
Department of CIS
PSG College of Technology
Coimbatore, India

R.Venkatesan
Professor and Head
Department of CSE
PSG College of Technology
Coimbatore, India

ABSTRACT
Web prefetching provides an effective mechanism to mitigate the user perceived latency when accessing the web pages. The content of web pages provide useful information for generating the predictions, which are used to prefetch the web objects for satisfying the user's future requests. In this paper, we propose fuzzy logic based web prefetching scheme that generates effective predictions for prefetching the web objects. Predictions are generated based on the anchor text information associated with hyperlinks in a web page. Based on the user's browsing pattern in each session, prediction engine dynamically computes the value and generates the list of predictions. The prefetched web objects are effectively utilized when user browses the web pages for information related to specific topic of interest. In long duration browsing sessions, useful predictions are generated to efficiently minimize the user perceived latency. The proposed scheme is compared with existing prefetching algorithms and the results indicate that the new scheme achieves improved cache-hit rate and precision accuracy.

General Terms
Web Mining, Prefetching

Keywords
Prefetching, Predictions, Fuzzy Logic, hyperlinks, Anchor text

1. INTRODUCTION

The exponential growth in usage of World Wide Web (WWW) and Internet for data dissemination often creates enormous traffic in the network causing noticeable Web page rendering delays when accessed by Web clients. Several factors (bandwidth availability, request processing time at server, round trip time, and object size) affect the user perceived latency when downloading a web page. Latency can be reduced by implementing cache repositories either remotely (in web server or proxy server) or locally (in browser's cache or local proxy server). The usage of cache repository is improved by applying Web prefetching mechanism that acquires Web contents with the anticipation that these contents will be requested by the users in the near future. Web prefetching exploits the benefit of spatial locality exhibited by the users when accessing the Web objects.

Fuzzy logic has been applied in several domains over the years such as expert systems, data mining and pattern recognition. It deals with fuzzy sets [1] that allow partial membership in a set represented by its degree of relevance. Fuzzy logic is capable of handling approximate or vague notions that exist in several information retrieval (IR) tasks [2] and helps to establish meaningful and useful relationships among objects. In this paper, we use fuzzy logic to compute the prediction value of hyperlinks and use it to select the appropriate hyperlinks to be prefetched. Client system decides to prefetch the Web objects based on the following factors [3]: availability of Web object in the cache and its current timestamp, user idleness for more than the threshold interval, network bandwidth, size of Web object and user preferences.

The proposed Web prefetching scheme uses the information associated with hyperlinks in the current Web page to predict the Web objects to be retrieved for satisfying user's future requests. Prefetching is carried out during idle time period between the user accesses to the Web pages. The prefetching component is attached to the Web browser to determine the Web objects to be prefetched. The working of proposed scheme has the following steps: 1) Extract the hyperlinks and its associated anchor text from the displayed Web page 2) Collect set of tokens (keywords) from each anchor text 3) Compute the prediction value for each hyperlink by applying fuzzy logic over the set of tokens 4) Generate the predictions (hint list) based on the computed prediction values 5) Prefetch the Web objects based on the links listed in the hint list. When user wishes to view a new Web page by clicking a hyperlink or typing URL in the Web browser, prefetch cache is first accessed to verify if it can satisfy the request before forwarding the request to proxy or Web server. Prediction value of each hyperlink is computed using the information stored in user-accessed and predicted-unused repositories. The information in predicted-unused repository is used to filter out hyperlinks that are of less or no interest to the user. The experimental results clearly indicate the efficiency of proposed Web prefetching scheme in enhancing the cache hit-ratio and significantly lowering the Web page access delay.

The rest of this paper is organized as follows: Section 2 discusses the related work in Web caching and prefetching. Section 3 discusses the architecture and working of proposed Web prefetching scheme. Section 4 presents the evaluation details and the observed results. Finally, section 5 concludes the paper.

2. RELATED WORK

There has been significant amount of research work carried out in the past for enhancing the performance of Web caching and prefetching. Several techniques were designed to be used at client-side, server-side and hybrid client/server for enhancing the delivery of Web pages to the user. User's browsing behavior was analyzed to identify specific interest on a domain for supporting services like Web personalization and prefetching. The effectiveness of using link-based or content-based ranking method in finding the Web sites was analyzed in [4] and the results indicated that anchor texts were highly useful in site finding. A text analysis method was discussed in [5] that used text in and around the hypertext anchors of selected Web pages to determine the user's interest in accessing the Web pages. In [6] a keyword-based semantic prefetching approach was proposed that applied neural networks to predict the future

requests based on the semantic preferences of past retrieved Web documents. A personalization algorithm was designed in [7] to combine the usage data and link analysis techniques for ranking and recommending the Web pages to the end user. A methodology to prefetch the Web objects of slower loading Web pages by semantically bundling it with faster loading Web pages was proposed in [8].

A Semantic Link Prefetcher was proposed [9] to utilize the semantic link information associated with the current Web page hyperlinks to predict the Web objects to be prefetched during the limited view time interval of the current Web page. In [10] a transparent and speculative algorithm was proposed for content based Web page prefetching with the assumption that textual information in both the visited pages and the followed links were influential in determining the preferences of a user. A novel non-intrusive Web prefetching system was presented in [11] to avoid the interference between prefetch and demand requests by effectively utilizing only the spare resources on the servers and network. The system was deployed without making any modifications to the Web browser, HTTP protocol and the network. In [12] a client-based Web prefetching system was proposed that used detection theory to determine the threshold value for selecting the Web documents to be prefetched.

The path profiles of users stored in large Web logs provide useful information for predicting the user's future requests. An n-gram based model [13] used an efficient method to compress the prediction model size to fit in main memory. It improved the prediction accuracy substantially with a moderate decrease in applicability. A history based prefetching algorithm [15] achieved high prediction accuracy with limited memory by storing only the useful request sequences and discarding those that will not yield useful predictions. In [14] a methodology was proposed to cluster related pages into different categories based on the access patterns. Pages were further categorized into levels based on the page rank and those pages that are placed in the top levels had higher probability of being predicted and prefetched. In [16] methods were proposed for modeling the user navigation history by extracting knowledge from user access sequences and Web page content.

PPM models were commonly used in Web prefetching for predicting the user's next request by extracting useful knowledge from historical user requests. Factors such as page access frequency, prediction feedback, context length and conditional probability influence the performance of PPM models in prefetching. An online PPM model based on non compact suffix tree was implemented in [17] that used maximum entropy principle to improve the prefetching performance. A novel PPM model based on stochastic gradient descent was presented in [18] that defined a target function to describe a node's prediction capability and then selected a node with maximum function value to predict the next most probable page.

Markov models were effectively used in Web prefetching by utilizing the information gathered from Web logs. In [19] different techniques were presented for intelligently selecting the parts of different order Markov models to create a new model with reduced state complexity and improved prediction accuracy. Three schemes of pruning (support, confidence and error) were presented to prune the states of All-K^{th} order markov model. A Markov–Knapsack approach was proposed in [20] that combined Multi-Markov Web-application centric prefetch model with a Knapsack Web object selector for enhancing the Web page rendering performance. An integration model was designed [21] to combine clustering, association rules and Markov models to achieve better prediction accuracy with minimal state space complexity. Markov tree [22] was used for effective page predictions and cache prefetching, which used the training data set to construct the tree structure for representing the Web page access patterns of users.

Domain ontology provides useful semantic information to be used in next page prediction systems. In [23] two methods were discussed to integrate the semantic information into Markov models for prediction. The methods allowed low order Markov models to make intelligent accurate predictions with less complexity than the higher order models. An approach for Web page prediction through linear regression was proposed in [24] that depended on the transition probability and ranking of links in the current Web page for the prediction accuracy.

To effectively reduce the user-perceived latency, prediction algorithm need to consider the structure of current Web pages when generating the predictions. This issue was addressed in [25] by developing a Double Dependency Graph (DDG) algorithm that differentiated HTML and embedded objects to create a new prediction model according to the structure of current Web. In [26] a decision method was presented to select the training data by monitoring the prediction precision, where the user access sequences were partitioned into different data blocks based on the access time of requests.

3. PROPOSED METHODOLOGY

The architecture of a basic Web system consists of *clients* that are used to access the Web and *Web servers* that respond to user's requests using the stored or generated information. In demanding situations, proxy server is deployed between clients and Web server to minimize Web server load and user access latency. Web prefetching can be integrated into the existing architecture by implementing two important components: Prediction and Prefetching engine. These two components can be deployed in any part of the Web architecture (i.e. client, proxy or server) to minimize the user perceived latency. Based on the deployment of prediction engine in the Web architecture, it uses different information to generate the predictions. When it is deployed at client, it uses information specific to a particular user. When deployed at proxy server, it uses information related to group of users. When deployed at Web server, it uses information from wide range of clients.

In the proposed Web prefetching scheme, both the prediction and prefetching engine are deployed at the client machine to minimize user perceived latency. Prediction engine uses the information associated with hyperlinks of a Web page to generate the predictions (i.e. hint list) and supply as input to the prefetching engine. Prefetching engine uses the hint list information to prefetch the Web objects and store it in the prefetch cache maintained at the client to satisfy the user requests.

Figure 1 illustrates the process of gathering relevant hyperlinks from the Web pages to generate the hint list for prefetching the Web objects. The proposed scheme is designed to efficiently identify set of hyperlinks in a Web page that reflect user's interest and prefetch them before the user actually requests those pages.

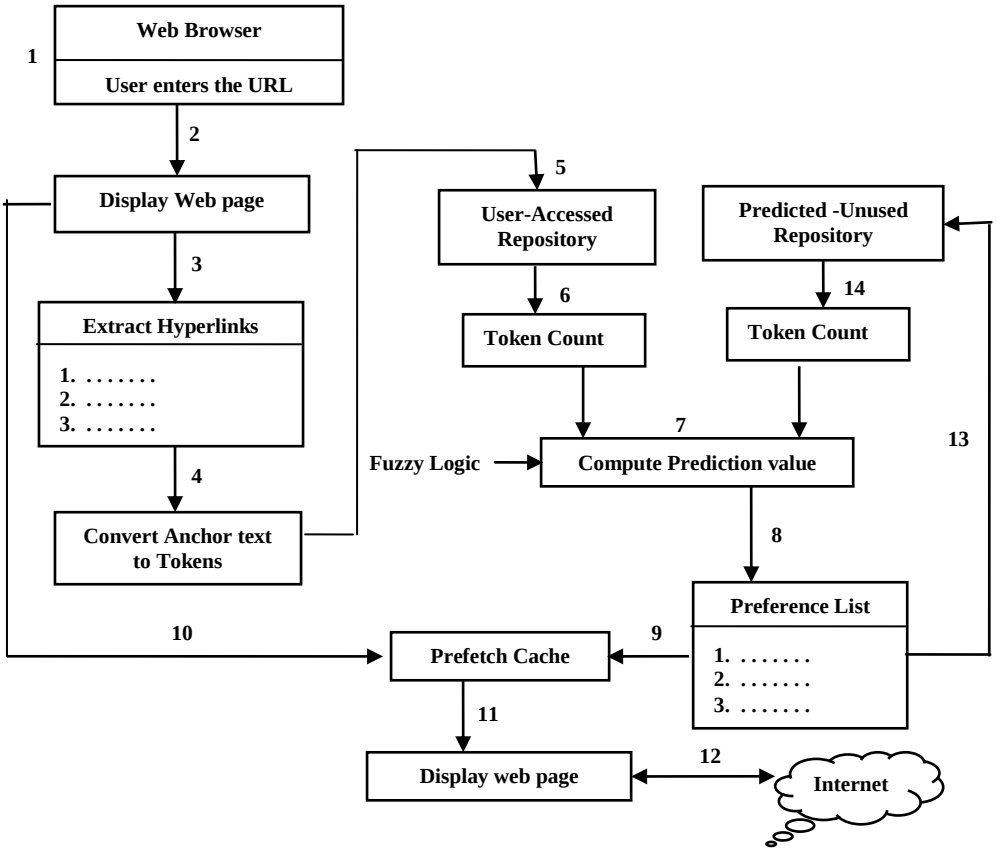

Figure 1: Process of Predicting and Prefetching the Web objects

The steps shown in Figure1 are explained as follows:

1. User initially requests a Web page by typing its URL in the Web browser.
2. The requested Web page gets displayed on the browser after its contents were downloaded from the Web server.
3. The displayed Web page is parsed to extract all the hyperlinks and its associated anchor text for computing the prediction value.
4. Each anchor text is treated as set of tokens, where token represents a meaningful word in the anchor text.
5. When user visits a new Web page by clicking hyperlink in the current Web page, then tokens of that anchor text are added to the user-accessed repository.
6. Count value of tokens gets updated in the user-accessed repository whenever user visits a new Web page by clicking hyperlinks.
7. Prediction value of each hyperlink is computed by applying fuzzy logic over the information stored in user-accessed and predicted-unused repository. Initially the predicted-unused repository remains empty and then it is loaded with tokens once the prediction activity is started.
8. Based on the computed prediction value, hyperlinks are sorted (highest to lowest) to create a hint list.
9. Prefetch engine retrieves Web objects from the server using the hyperlinks in the hint list and stores them in the prefetch cache.
10. When user wishes to view a new Web page by clicking hyperlink in current Web page or typing URL in the Web browser, contents of prefetch cache will be verified to check if they can satisfy the user request.
11. When the user requested Web page is available in the prefetch cache, then it gets displayed in the Web browser with minimal latency.
12. In case the requested Web page is not available in the prefetch cache, then it will be retrieved from the Web server and displayed to the user.
13. Tokens of hyperlinks in the hint list that were not used by the users will be moved to predicted-unused repository.
14. When user visits a new Web page, hint list will be populated with new set of hyperlinks using which the prefetching activity is carried out. Count value of tokens gets updated in the predicted-unused repository whenever unused links are identified and their tokens moved to the repository.

3.1 Prediction Engine

It is responsible for computing the prediction value of hyperlinks by applying fuzzy logic over the set of tokens related to hyperlinks. The prediction engine uses tokenizer and

repositories (user-accessed and predicted-unused) in computing the prediction value and creating the hint list that is given as input to the prefetching engine.

3.1.1 Tokenizer

When user views a Web page in the browser, the Tokenizer parses that Web page to extract all the hyperlinks (URL) and its associated anchor text. Anchor text refers to the text that surrounds hyperlink definitions (hrefs) in Web pages [6]. In our scheme, we consider the text between the tags <a> and . Each anchor text is represented as set of tokens, where token is a meaningful word within the anchor text of a link. When user clicks a hyperlink in the current Web page to view the next Web page, then the tokens of its anchor text are stored in the user-accessed repository. Tokenizer eliminates trivial characters and tokens such as prepositions and conjunctions present in the anchor text and consider only the meaningful words as tokens and store it in user-accessed repository. Stemming is applied over the tokens of anchor text to eliminate the word suffixes, so that the number of entries in the repository gets minimized. The commonly used Porter stemmer [27] algorithm is applied to perform the stemming operation.

3.1.2 User-Accessed repository

It maintains tokens with their occurrence count that gets incremented each time it is present in the anchor text of hyperlinks used by the user. Each token has an initial count value of 1. The information maintained in this repository is used for computing the prediction value of each hyperlink. The repository reveals both user and session characteristics, where a session indicates the time interval between start and end of user's browsing instance.

3.1.3 Predicted- Unused Repository

The tokens of hyperlinks that are predicted but not used by the users are stored in this repository. It provides feedback to the prediction engine that helps to refine the hint list generated from the Web pages. The repository provides two major benefits: a) Minimize the influence of tokens in the user-accessed repository that are of less or no interest to the user when computing prediction value b) Minimize the number of predictions generated per page. For each Web page 'n' number of predictions are generated, but when the user navigates from current Web page to the next page only one from the prediction list will match the user's interests. The remaining hyperlinks in the hint list do not reflect the user's interests and their tokens are stored in this repository.

The process of adding tokens to the predicted-unused repository are:

1. The prediction engine recommends set of hyperlinks (hint list) for a Web page based on the computed prediction values.
2. From the recommended set of hyperlinks, tokens are collected and stored in a temporary buffer.
3. Check if the hyperlink used to navigate from the current Web page to the next page matches with any of the hyperlink in the hint list. If there is a match, go to step 4 else step 6.
4. The tokens of hyperlink that match with the user used hyperlink are moved from the temporary buffer to the user-accessed repository.
5. The tokens of unmatched hyperlinks are moved from the temporary buffer to the predicted-unused repository. To create new hint list for the next web page, go to step 1.
6. All the tokens stored in the temporary buffer are moved to the predicted-unused repository. To create new hint list for the next web page, go to step 1.

3.1.4 Implementing Repositories

The repositories (user-accessed and predicted-unused) are implemented as a table with three fields: token, token-count and last-updated time. Token field is used to store the tokens extracted from the anchor texts. Token-count field indicates the number of times each token is updated in the repository. Last-updated time field indicates the time when the token was last added to the repository. Both the repositories are of fixed size and new tokens are added into it by eliminating the old tokens whenever the repositories reach their maximum limit. Based on the last-updated time field, the tokens are selected for elimination from the repositories. The repository size should be selected carefully in order to avoid the legitimate tokens from being eliminated and to prevent the trivial tokens from occupying the allotted space. We set the size of each repository to be 100.

3.1.5 Computing Prediction Values

The anchor text is represented as set of tokens arranged in a specific order.

Anchor text = $\{T_1, T_2, T_3 \ldots T_n\}$, n = number of tokens

The system is modeled by applying fuzzy logic over the tokens of anchor text by associating it with a fuzzy set (i.e. repository holding the tokens). Tokens of anchor text are related to fuzzy set with similarity degree in the range 0 to 1.

The membership value of each token T_i is computed by dividing the token count in repository R_1 with sum of token count in the repositories R_1(user-accessed repository) and R_2(predicted-unused repository).

$$\mu R_1(T_i) = \frac{(TC_i)R_1}{(TC_i)R_1 + (TC_i)R_2}$$

The membership value of token T_i relative to repository R_2 will be:

$$\mu R_2(T_i) = 1 - \mu R_1(T_i) \qquad [i = 1 \text{ to } n \mid n = \text{no. of tokens}]$$

The membership value of token T_i will be 1, if the token is available in only a single repository (i.e. R_1 or R_2).

$\mu R_1(T_i)$ = Membership of token T_i relative to repository R_1

$\mu R_2(T_i)$ = Membership of token T_i relative to repository R_2

$(TC_i)R_1$ = Token Count of T_i in repository R_1

$(TC_i)R_2$ = Token Count of T_i in repository R_2

After computing the membership value of each token T_i relative to the repositories R_1 and R_2, the values are compared to decide whether token T_i is included for computing the prediction value of a hyperlink.

$$\text{If } \mu R_1(T_i) > \mu R_2(T_i)$$

then

$TA_i = 1$

else

$TA_i = 0$

The value of TA_i will be 1, when the membership value of token T_i relative to R_1 is greater than R_2; else the value of TA_i will be 0. For token T_i with $TA_i = 1$, compute token popularity (TP_i) in the repository R_1 by dividing its count value with maximum token count value in R_1.

$$TP_i = \frac{(TC_i)R_1}{\max [(TC)R_1]}$$

For token T_i with $TA_i = 0$, its token popularity (TP_i) will be

zero.

Finally, the prediction value (PV) of hyperlink will be

$$PV = \frac{1}{n} \left[\sum_{i=1}^{n} TP_i \right]$$

where TA_i = Token Acceptance

n = no. of valid tokens in the anchor text

3.1.6 Hint List

Based on the computed prediction value, the hyperlinks are selected to create a hint list. The hint list is managed using a priority queue that arranges the hyperlinks according to its prediction value (highest to lowest). Prefetching engine uses the hyperlinks stored in the hint list to prefetch the Web objects during browser's idle time period. When user visits a new Web page, the contents of hint list will be cleared and then populated with new set of hyperlinks to perform prefetching. It helps to prevent prefetching of irrelevant hyperlinks during a browsing session. Figure 2 illustrates the process of adding hyperlinks to the hint list.

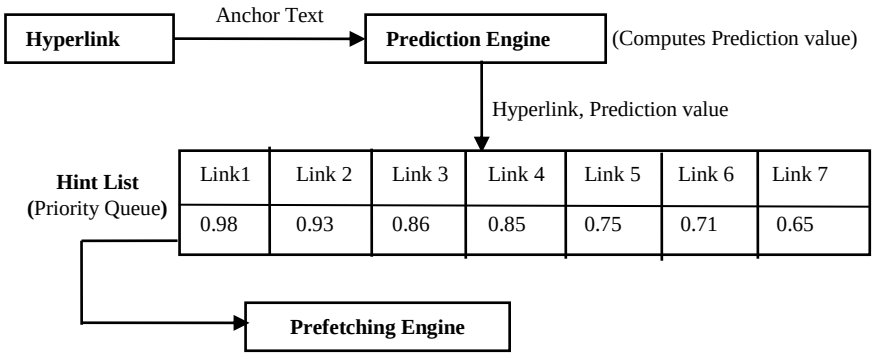

Figure 2: Creation of Hint List for Prefetching

3.2 PREFETCHING ENGINE

It uses the hyperlinks listed in the hint list to prefetch the Web objects during browser idle time to avoid interference with regular user requests. When a user requested Web object is available in the prefetch cache, then it is served quickly with minimal retrieval time. The prefetching engine will not retrieve all the predicted Web objects from the server due to the following reasons: a) Lack of idle time due to faster navigation between the Web pages by users b) Few predicted objects already exists in the regular cache and c) Few predicted objects already demand requested by the users. The prediction engine needs to generate useful hint list to avoid wastage of user and server resources that may lead to performance degradation. Prefetch requests are given low priority than the regular user requests to allow the Web browser to utilize the entire available bandwidth for satisfying the user requests. Whenever user initiates any page loading activity in the browser, prefetching activity gets terminated and the remaining information in the hint list will be discarded.

The downloaded Web objects from the server are stored in the prefetch cache for satisfying the user's future requests. Prefetch cache is implemented separately from the browser's regular cache to eliminate the caching effect due to temporal locality in the user's browsing patterns. LRU algorithm is used to manage the Web objects stored in the prefetch cache by selecting least recently accessed Web objects for purging to provide space for accommodating newly prefetched Web objects. When a demand requested Web object resides in prefetch cache, then it is moved to the regular cache. Web objects will not be stored in both the caches (regular and prefetch) at the same time.

4. EVALUATION

The performance of proposed prefetching scheme cannot be evaluated using trace based simulations, because the Web traces will not give a comprehensive view of the client's browsing interests. An effective way to gather the required information will be to capture the user's interest at the client side by analyzing the browsing pattern in each session. User can view the Web page in two ways: a) Type the required URL in Web browser and b) Click hyperlinks present in the current Web page. The proposed scheme extracts user's interests by observing the information associated with hyperlinks used to view the Web pages in each browsing session.

Evaluation of the proposed scheme is carried out by performing Web browsing that focuses on information specific to particular topic of user's interest. Experimentation carried out using open source Web browser- CXBrowser [28] developed in c# language. The prediction/prefetching engine implemented as an add-on to the Web browser that allows user to configure the

prefetch settings based on the requirements. Each browsing session has active and idle periods, where the user switches between them based on his access pattern. Active period indicates the phase where the Web objects are demand requested by the user. Idle period indicates the phase where the displayed Web objects are viewed by user. The retrieval of main html file followed by its embedded objects initiates the active period. In idle period, prefetching engine uses the hint list to prefetch Web objects and stores it in the prefetch cache. Log file is used to record the user's requests during the browsing session, which is used for analyzing the system performance.

When user initially starts the browsing session, user-accessed and predicted-unused repositories remain empty and they cannot be used to predict the Web objects to be prefetched. The user-accessed repository will be filled with tokens as the user starts browsing Web pages using the hyperlinks. The predicted-unused repository will be filled with tokens of hyperlinks that are predicted but not used either to prefetch the Web objects or to satisfy the user's future requests. The user navigation time partitioned into four discrete intervals [29]: passing, simple viewing, normal viewing and preferred viewing. The amount of time user spends reading a Web page influence the number of links that can be prefetched. It was considered an important attribute in predicting the user's interests [30, 31, 32]. If user spends more time reading a Web page, browser idle time will be more allowing several hyperlinks in the hint list to be prefetched.

When a Web page viewed by the user contains information that is less relevant to his/her interests, then hint list for that Web page contains zero or minimal number of hyperlinks. Web objects that are demand requested by the user will be first verified in the local cache (regular or prefetch cache) for its availability. Web or proxy server is contacted only when the local cache does not hold the requested Web objects.

4.1 Experimental Results

The performance of proposed Web prefetching scheme depends on the reading interest of individual users and hence the results from different users are incomparable. The results are taken by observing several browsing sessions carried out for a period of six weeks. Performance of the proposed scheme is compared with Top-down approach [34], Naïve-Bayes approach [33] and Bigrams in Link approach [10]. Cache hit rate and Accuracy are the metrics used for evaluation.

Cache hit-rate indicates the percentage of user requests served using the contents of prefetch cache against the total number of user requests. High hit-rate effectively minimizes the user perceived latency, since most of the user requests are served from the local cache.

$$\text{Hit Rate} = \frac{\text{Prefetch Hits}}{\text{Total User Requests}}$$

Accuracy indicates the percentage of prefetched Web pages requested by the user against the total number of Web pages prefetched by the system. It reflects the useful Web predictions generated during the browsing sessions.

$$\text{Accuracy} = \frac{\text{Prefetch Hits}}{\text{Total Prefetchs}}$$

The comparison of cache hit-rate for various schemes is shown in Figure 3. The number of links prefetched per page using the predictions is varied between 2 to 10. When more links are prefetched, it improves the hit-rate significantly. The proposed scheme achieves better hit-rate in all the cases, since it can fine tune the predictions based on the access pattern of user. It effectively filters out the unwanted hyperlinks when generating the predictions using the information from predicted-unused repository. Tokens of anchor text that has higher presence in the user-accessed repository than predicted-unused repository are the ones used for computing the prediction value of hyperlinks.

Figure 4 shows the comparison of prediction accuracy for various prefetching schemes. Prefetching two links per page could not satisfy the user requests effectively, resulting in minimal usage of prefetched pages during the browsing sessions. When the number of links prefetched per page varies between 4 and 6, the contents were effectively used to satisfy the user requests. When eight or more links are prefetched per page, it improves the hit rate but it leads to prefetching of unwanted links that wastes network resources. The proposed scheme outperforms other approaches in accurately generating the predictions.

5. CONCLUSION

In this paper we have discussed Web prefetching scheme that used fuzzy logic to compute the prediction value of hyperlinks, which was used to decide the Web objects to be prefetched. Information available in the user-accessed and predicted-unused repositories was used to compute the prediction value of hyperlinks, which improved the prediction accuracy and minimized user perceived latency. It generated effective predictions during the browsing sessions, when user visited Web pages seeking information relevant to specific topic of interest. The proposed scheme was experimentally evaluated by observing the results over several user browsing sessions. Results indicate that the proposed scheme provides good hit rate and precision accuracy, when compared to other existing algorithms.

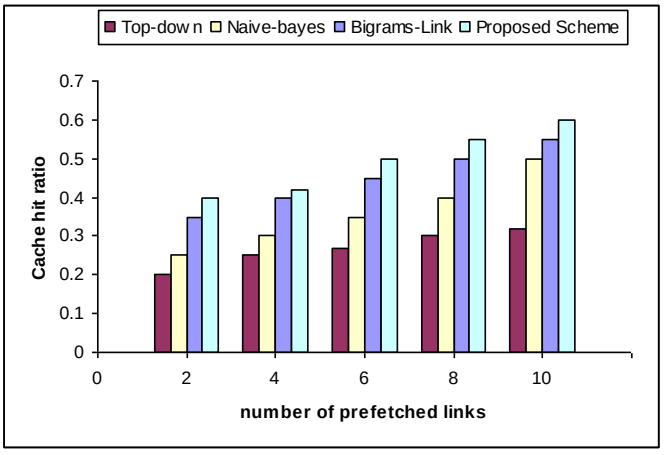

Figure 3: Hit-Rate of various prefetching schemes

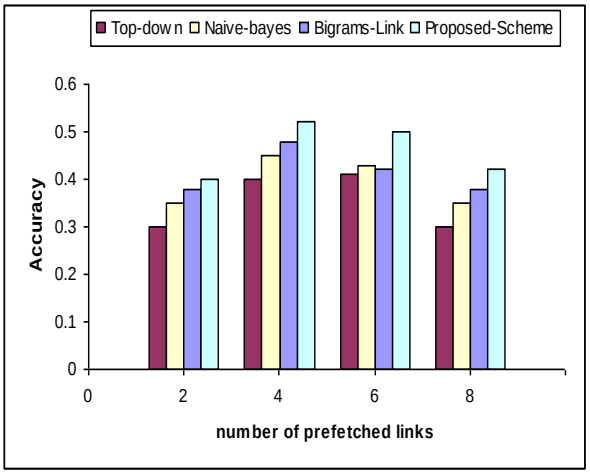

Figure 4: Accuracy of various prefetching schemes

6. REFERENCES

[1] L.A.Zadeh, "Fuzzy Sets", Information and Control, Vol.8, pp.338-353, 1965

[2] H. Chris Tseng, "Internet Applications with Fuzzy Logic and Neural Networks: A Survey", Journal of Engineering, Computing and Architecture, Volume 1, Issue 2, 2007

[3] J.C.Mogul, "Method for predictive prefetching of information over a communications network", Patent No.5,802,292, 1998

[4] N. Craswell, D. Hawking, S.E. Robertson, "Effective Site Finding Using Link Anchor Information", Proceedings of International ACM SIGIR Conference on Research and Development in Information Retrieval, 2001

[5] B.D.Davison, "Predicting web actions from HTML content", Proceedings of 13th ACM Conference on Hypertext and Hypermedia, 2002

[6] Cheng-Zhong Xu and Tamer I. Ibrahim, "A Keyword-Based Semantic Prefetching Approach in Internet News Services", IEEE Transactions on Knowledge and Data Engineering, Vol. 16, No. 5, pp.601 -611, 2004

[7] Magdalini Eirinaki, Michalis Vazirgiannis, "Usage-based PageRank for Web Personalization", In proceedings of 5th IEEE International Conference on Data Mining (ICDM), 2005

[8] Alexander P. Pons, "Semantic prefetching objects of slower web site pages", The Journal of Systems and Software, Vol.79, pp.1715–1724, 2006

[9] Alexander P. Pons, "Object Prefetching Using Semantic Links", ACM SIGMIS Database, Vol.37 Issue 1, pp. 97 – 109, 2006.

[10] A. Georgakis, H. Li, "User behavior modeling and content based speculative web page prefetching", Data and Knowledge Engineering - Elsevier, vol.59, pp.770 - 788, 2006

[11] Ravi Kokku, Praveen Yalagandula, Arun Venkataramani and Michael Dahlin, "NPS: A non-interfering deployable web prefetching system", In Proceedings of the USENIX Symposium on Internet Technologies and Systems, Palo Alto, USA, 2003

[12] Kelvin Lau, Yiu-Kai Ng, "A Client-based Web Prefetching Management System Based on Detection Theory", Lecture Notes in Computer Science- Springer, vol. 3293, pp. 129-143, 2004.

[13] Zhong Su, Qiang Yang, Hong-Jiang Zhang, "A Prediction System for Multimedia Pre-fetching in Internet", in Proceedings of the eighth ACM international conference on Multimedia, pp. 3 – 11, 2000

[14] Debajyoti Mukhopadhyay, Priyanka Mishra, Dwaipayan Saha, Young-Chon Kim, "A Dynamic Web Page Prediction Model Based on Access Patterns to Offer Better User Latency" , in Proceedings of the 6th International Workshop (MSPT- 2006), pp. 59–64, November 2006

[15] Qinghui Liu, Roberto Solis-Oba, "Web Prefetching with High Accuracy and Low Memory Cost", Applied Computing Conference (ACC '08), Istanbul, Turkey, May 27-30, 2008.

[16] Costantinos Dimopoulos, Christos Makris, Yannis Panagis, Evangelos Theodoridis,Athanasios Tsakalidis, "A web page usage prediction scheme using sequence indexing and clustering techniques", Data & Knowledge Engineering-Elsevier, vol.69, pp.371–382, 2010

[17] Zhijie Ban, Zhimin Gu, and Yu Jin, "An online ppm prediction model for web prefetching", In proceedings of the 9th annual ACM international workshop on Web information and data management, Lisbon, Portugal, 2007.

[18] Zhijie Ban, Zhimin Gu, and Yu Jin, "A PPM prediction model based on stochastic gradient descent for web prefetching", In proceedings of the 22nd International Conference on Advanced Information Networking and Applications, Okinawa, Japan, 2008.

[19] Mukund Deshpande and George Karypis, "Selective Markov Models for Predicting Web Page Accesses", ACM Transactions on Internet Technology, Vol. 4, No. 2, pp.163-184, May 2004

[20] Alexander P. Pons, "Improving the performance of client web object retrieval", Journal of Systems and Software, vol.74, No.3, 2005.

[21] Khalil, F., Li, J. and Wang, H., "An integrated model for next page access prediction", International Journal of Knowledge and Web Intelligence, Vol. 1, Nos. 1/2, pp.48–80, 2009

[22] Wenying Feng, Shushuang Man and Gongzhu Hu, "Markov Tree Prediction on Web Cache Prefetching", In Proceedings of Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, pp. 105-120, 2009.

[23] Nizar R. Mabroukeh and C. I. Ezeife, "Semantic-rich Markov Models for Web Prefetching", in proceedings of IEEE International Conference on Data Mining Workshops, 2009

[24] Ruma Dutta, Anirban Kundu, Rana Dattagupta, Debajyoti Mukhopadhyay, "An Approach to Web Page Prediction Using Markov Model and Web Page

Ranking", International Journal of Convergence Information Technology, Korea, Vol.4, No.4, pp. 61–67, December 2009.

[25] J. Domenech, J.A. Gil, J. Sahuquillo, A. Pont, "Using current web page structure to improve prefetching performance", Computer Networks, vol.54, pp.1404–1417, 2010.

[26] Zhijie Ban, Feilong Bao, "Decision Method of Training Data for Web Prefetching", in proceedings of the Sixth International Conference on Internet and Web Applications and Services (ICIW), 2011

[27] M.F. Porter, An algorithm for suffix stripping, Program, vol.14, no.3, pp.130–137, 1980

[28] CxBrowser – http://cxbrowser.sourceforge.net

[29] Dongshan Xing and Junyi Shen, "Efficient data mining for web navigation patterns", Information & Software Technology, vol.46, no.1, pp.55–63, 2004

[30] Ting-Peng Liang and Hung-Jen Lai, "Discovering User Interests from Web Browsing Behavior: An Application to Internet News Services", Proceedings of the 35th Hawaii International Conference on System Sciences, 2002

[31] S. Gunduz and M. Ozsu, "A web page prediction model based on click-stream tree representation of user behavior", In Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 535–540, 2003

[32] Yong Zhen Guo, Kotagiri Ramamohanarao and Laurence A. F. Park, "Personalized PageRank for Web Page Prediction Based on Access Time-Length and Frequency", In proceedings of IEEE/WIC/ACM International Conference on Web Intelligence, 2007

[33] P.Venketesh, R, Venkatesan, L.Arunprakash, "Semantic Web Prefetching Scheme Using Naïve Bayes Classifier", in International Journal of Computer Science and Applications, Vol. 7, No. 1, pp. 66 – 78, 2010

[34] E.P. Markatos, C. Chronaki, "A top-10 approach to prefetching on the Web", in Proceedings of INET'98, 1998.