Journal of Web Engineering, Vol. 13, No.5&6 (2014) 507-524
© Rinton Press

WEB PAGE PREDICTION ENHANCED WITH CONFIDENCE MECHANISM

ARPAD GELLERT

Computer Science and Electrical Engineering Department, Lucian Blaga University of Sibiu, Romania
arpad.gellert@ulbsibiu.ro

ADRIAN FLOREA

Computer Science and Electrical Engineering Department, Lucian Blaga University of Sibiu, Romania
adrian.florea@ulbsibiu.ro

Received October 29, 2013
Revised July 22, 2014

In this work we comparatively present and evaluate different prediction techniques used to anticipate and prefetch web pages and files accessed via browsers. The goal is to reduce the delays necessary to load the web pages and files visited by the users. We have included into our analysis Markov chains, Hidden Markov Models and graph algorithms. We have enhanced all these predictors with confidence mechanism which classifies dynamically web pages as predictable or unpredictable. A prediction is generated only if the confidence counter attached to the current web page is in a predictable state, improving thus the accuracy. Based on the results we have also developed a hybrid predictor consisting in a Hidden Markov Model and a graph-based predictor. The experiments show that this hybrid predictor provides the best prediction accuracies, an average of 85.45% on the "Ocean Group Research" dataset from the University of Boston and 87.28% on the dataset collected from the educational web server of our university, being thus the most appropriate to efficiently predict and prefetch web pages.

Key words: Web page prediction, prefetching, Markov chains, Hidden Markov Models, graph algorithms, browser extension.
Communicated by: B. White & O. Pastor

1 Introduction

Nowadays the access to the Internet becomes more prevalent worldwide and often there are requests for higher bandwidths. According with Yahoo [2], today, there are over 130 million Web servers and the Web is the largest data repository (estimated to 100 billion pages). By the end of 2011 the International Data Corporation predicted that the amount of information in the world will exceed 1.8 zettabytes (2^{70}). The amount of archived web pages exceeds two petabytes (2^{50}) and it increases at a rate of 20 terabytes (2^{40}) each month. These statistics are expressed in other words by Eric Schmidt, the executive chairman of Google, namely, the digital world "now produces as much information in two days as it did all the way along from the dawn of man up to the year 2003" [25]. Also, current economic companies rely on technology and Internet communications, and information came to travel and appear at a fantastic rate. However, only a small part of the information conveyed is relevant to the user at a time. Therefore, creating a user profile based on recent history of visited pages can reduce waiting times.

Clients are frequently confronted with delays in accessing web pages, especially those ones that are limited by low-bandwidth modems or wireless routers. Many latency-tolerant techniques have been developed during the last years, the most important being caching and prefetching. Caching exploits the temporal locality principle by keeping the accessed pages and files in a cache structure, whereas prefetching anticipates the next accesses and loads the corresponding web pages or files into the cache. If the user accesses a web page or a file which is available in the cache, the browser can load it without any delays. Thus, when the users have long browsing sessions, prediction-based prefetching can be very effective by minimizing the access latencies.

In this paper we analyze comparatively different web page prediction techniques based on Markov chains, Hidden Markov Models (HMM) and graphs, all of them enhanced with a confidence mechanism. We propose also a hybrid predictor. The general applicability of the techniques considered confers a greater portability to our methods. We evaluate the mentioned predictors in terms of prediction accuracy. The goal is to find the most appropriate prediction technique for anticipating and prefetching web pages and to integrate it as an extension into browsers.

The organization of the rest of this paper is as follows. In Section 2, we are reviewing the related work in web page prefetching and also in prediction algorithms. Section 3 is describing our proposed web page predictors. In Section 4 we are presenting the experimental results. Conclusions and further work directions are included in Section 5.

2 Related Work

In our previous works [30, 11] we already implemented multi-layer perceptrons, Markov chains and HMMs for next location prediction, to anticipate the movements of employees within an office building by predicting the next rooms based on the history of visited rooms. The goal of the research was to design some smart doorplates that are able to direct visitors to the current location of an office owner based on a location-tracking system and predict if the office owner is coming back soon. These predictors were evaluated on some movement sequences of real persons, acquired from the Smart Doorplates project developed at the University of Augsburg [23]. The experimental results showed that the accuracy in the next location prediction reaches up to 92%.

In his work [24] Rabiner showed how HMMs can be applied in speech recognition. The author presented the theory of HMMs from the simplest concepts (discrete Markov chains) to the most sophisticated models (variable duration, continuous density models, etc.). He also illustrated some applications of the theory of HMMs to simple problems in speech recognition, and pointed out how the techniques were applied to more advanced speech recognition problems. In [31] we used, among other metrics, a HMM-based prediction algorithm to evaluate the random degrees of some difficult to predict branches and we showed that these branches have intrinsic random behavior, being generated by very complex program structures. In [16] the authors used HMMs for semantic-based web page prefetching. In contrast, we do not use web semantics, we predict web pages based only on the history of links, exploiting pattern repetition. We also use a confidence mechanism in order to increase accuracy.

In [14] the authors proposed a continuous-time Markov model for web page prediction. They used Kolmogorov's backward equations to compute the transition probabilities and to predict which web page will be visited by a certain user and when. Their method can also estimate the total number of visits to a web page by all the people at a certain interval. Link prediction based on Markov chains was presented also in [33, 7, 26]. In [17] the author applied the Markov model together with the k-nearest neighbor classifier algorithm to enhance the performance of traditional Markov chains in predicting web pages. He obtained lower consumed memory, quite similar build time and evaluation time and higher accuracy. In [18, 19] clustering, association rules and lower order Markov models were combined providing an improved prediction accuracy and state space complexity. Clustering is used to group homogeneous user sessions, low order Markov chains are built on these clustered sessions and when Markov models cannot make clear predictions the association rules are used. The proposed integrated model has less state complexity and is more accurate than a higher order Markov model. In [20] the authors presented a survey of web page ranking for web personalization. They concluded that low order Markov models have higher accuracy and lower coverage, whereas the higher order models have a number of limitations associated with: higher state complexity, reduced coverage and sometimes even worse prediction accuracy.

Graphs were also used for prediction through some specific algorithms. In [15] Huang explored link prediction based on graph topology. Hasan addressed link prediction in social networks by using supervised learning and classification algorithms such as decision tree, multi-layer perceptron, support vector machines [13]. His results were reported on BIOBASE and DBLP, two datasets that have information about different research publications in biology and computer science, respectively. Unlike Hasan's work, we have simulated on the "Ocean Group Research" benchmarks from the University of Boston (BU) [6] and on the dataset collected from the educational web server of "Lucian Blaga" University of Sibiu (LBUS) and our algorithms are not applied strictly on social networks but are useful for predicting any web pages. We have also used a confidence mechanism in order to increase accuracy.

Another approach in link prediction of social networks was proposed in [22]. The author's solution aims at predicting links based on weighted proximity measures of social networks relying on the structural properties (topology) of a given network. There are some similarities between our work and [22], namely: the prediction is based on weighted graphs and we reached the same conclusion that considering the recently used link can be more appropriate in next page prediction. However, the calculation of weighted graphs and the data sets are different. Also, the solution from [22] has a particular character being suitable only for open and dynamic online social networks.

In [12] the authors presented a web page prediction method based on conditional random fields, used for labeling and segmenting sequential data. Conditional random fields have the ability to model the dependencies among observations and they can also incorporate various features from observation sequences to increase the prediction accuracy. They concluded that conditional random fields outperformed Markov chains and HMMs but for a great number of labels their training may become very expensive and even intractable. Their results also showed that the second order Markov chains and HMMs are better than the first order ones. In contradiction, our experiments show that the first order Markov chains and HMMs outperform the second order ones. An explanation for this difference can be the different data sets, as they used *msnbc*, whereas we used the BU and LBUS datasets.

In [10] the authors proposed a hybrid web page prediction method which combines support vector machines, association rule mining and Markov chains in order to enhance the efficiency. Their experimental results showed that the proposed hybrid predictor outperformed the individual predictors.

In [28] the authors presented an *n*-gram based model to utilize path profiles of users from very large web log to predict future requests. They showed that the proposed method achieves a reasonable balance between precision and applicability.

In [4] the authors proposed a page rank algorithm to predict the next page that will be visited by a web surfer. For the first set of sessions they applied the page rank algorithm which provides the ranks for web pages. For each web page their method determines to which pages the user can navigate and, using the page ranks, it computes the probability of visiting them by dividing each rank to the sum of ranks, and the number of links to the total number of links, respectively. The authors evaluated the proposed algorithm on the NASA dataset with an accuracy improvement from 19.75% to 32.5% and also on a log file from a commercial web site with an accuracy improvement from 74.66% to 77.77%.

In [3] Canali et al. proposed adaptive algorithms that combine predictive and social information and dynamically adjust their parameters according to the continuously changing workload characteristics. Their experimental results showed that such adaptive algorithms can achieve performance close to theoretical ideal algorithms.

Some of the algorithms proposed in the literature consider a training period before making predictions. The training period can improve or even decrease performance, since, if it is too long, it can involve a high amount of resources. In [9] the authors analyzed how this training affects the prediction accuracy.

In [32] Wan et al. proposed an approach based on a vector space model, called random indexing, for the discovery of the intrinsic characteristics of web user activities. The underlying factors are then used for clustering individual user navigational patterns. The clustering results are used to predict and prefetch web requests for grouped users.

In [29] Temgire et al. presented a review on web prefetching techniques. The huge variety of prefetching algorithms hinder to a certain extent the performance evaluation of new techniques and their correct comparison with existing methods. Therefore, in [8] the authors proposed a free environment for implementation and efficient evaluation of prefetching techniques. Their framework combines real and simulated behavior of web users and servers.

3 Prediction of Web Pages

The general prediction mechanism consists in anticipating future contexts based on current and previous context information, recovering the correct context if the speculation fails and updating the predictor to improve future prediction accuracy. Prediction can be very useful if the availability of some data in advance allows to reduce waiting times, improving thus the efficiency. Obviously, the prediction must be accurate, because in some applications a misprediction leads to certain costs due to the necessity of correct state recovery.

For predicting or anticipating future situations, some learning techniques like Markov chains, HMMs, graph algorithms, bayesian networks, time series or neural networks are obvious candidates. The challenge is to adapt such algorithms to work with context information.

As Figure 1 illustrates, our proposed browser extension, when activated, collects the links accessed by the user which are then preprocessed: each link is codified with a unique number, ports and relative parts are eliminated from links, if there are two consecutive accesses of the same link only one is considered, links having the extension *.gif* and *.xbm*, which are usually small images, are also eliminated. The browser extension keeps a certain history of the accessed links. When the current link is accessed, the next link is predicted on the basis of the history of the previously visited links. The predicted web page or file is prefetched in the cache in order to be available if the user accesses it.

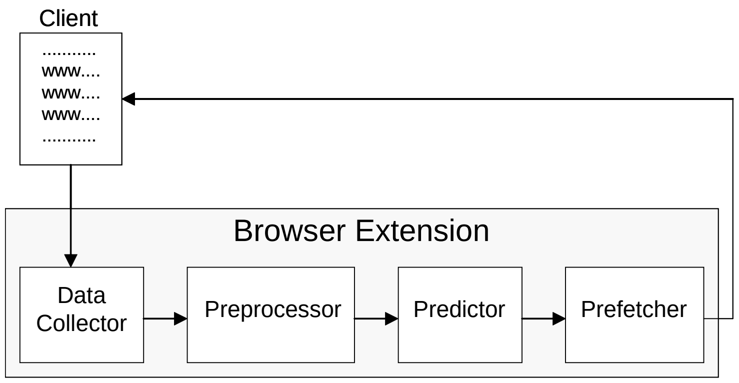

Figure 1. The structure of the application.

In order to improve the prediction accuracy, the predictor can be enhanced with a confidence mechanism which consists in saturating counters, attached to all the links kept in the history, which are incremented on correct prediction and decremented on misprediction. A prediction is generated only if the confidence counter of the current link is in a predictable state. A possible confidence automaton is presented in Figure 2.

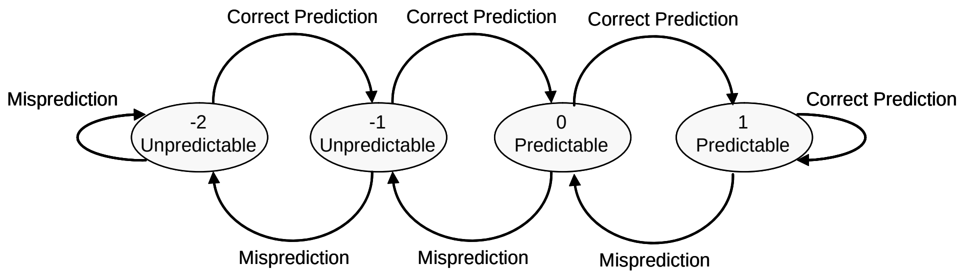

Figure 2. Confidence automaton with 4 states: 2 unpredictable and 2 predictable.

By using such a confidence mechanism, the number of predictions is lower but the prediction accuracy is significantly higher.

The next sections tackle with the way of we have adapted Markov chains, HMMs and graph algorithms to predict and prefetch web pages.

3.1 Markov Predictors

A first order discrete Markov process may be described at any time as being in one of a set of N distinct states $S = \{S_1, S_2, ..., S_N\}$ [24], as illustrated in Figure 3.

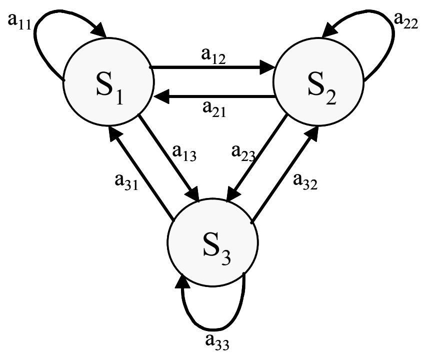

Figure 3. A Markov chain with 3 states.

The full probabilistic description of a discrete Markov chain requires the specification of the current state as well as all the predecessor states (the current state in a sequence depends on all the previous states). For the special case of a discrete, first order, Markov chain, this probabilistic description is truncated to just the current and the predecessor state (the current state depends only on the previous state): $P[q_t = S_j | q_{t-1} = S_i,\ q_{t-2} = S_k,\ ...] = P[q_t = S_j | q_{t-1} = S_i]$, where q_t is the state at time t. Thus, for a first order Markov chain with N states, the set of transition probabilities between states S_i

and S_j is $A = \{a_{ij}\}$, where $a_{ij} = P[q_t = S_j | q_{t-1} = S_i]$, $1 \le i,\ j \le N$, having the properties $a_{ij} \ge 0$ and $\sum_{j=1}^{N} a_{ij} = 1$. For a Markov chain of order *R*, the probabilistic description is truncated to the current and *R* previous states (the current state depends on *R* previous states).

The Markov chains can be used to predict the next value in a sequence based on a particular stored pattern. The prediction supposes searching for the context within the value sequence and determining which value followed the context with the highest frequency, that value being the predicted one. In a Markov chain of order *R*, the context consists in the last *R* values of the sequence, and, therefore, the search is done using this pattern of *R* values. A transition-table-based implementation is also possible, but it is inefficient for a high number of observation symbols. Theoretically, Markov chains can be used to predict any stochastic repetitive sequences.

The following example shows the necessity of using superior order Markov models. If the sequence of states is AAABCAAABCAAA, the Markov models of order 1 and 2 mispredict A, and only a Markov model of order 3 predicts the next state B correctly.

In our multi-page prediction approach we prefetch all the pages that appeared in the history after the considered context. The predictor can be enhanced with confidence mechanism (a prediction is generated only if the confidence counter of the current link is in a predictable state, see Figure 2).

3.2 HMM-Based Predictors

For the prediction of the next accessed web page we consider HMMs like those developed in [24, 11]. A HMM is a doubly embedded stochastic process with a hidden stochastic process that can only be observed through another set of stochastic processes that generate the sequence of observable symbols. A generic HMM is illustrated in Figure 4, where q_t is the hidden state at time *t*, O_t is the observation at time *t*, *A* is the matrix of transition probabilities between hidden states, and *B* is the matrix of observation probabilities within each hidden state.

$$\begin{tikzcd}
\text{Hidden State Sequence (Q):} & q_1 \arrow[r, "A"] \arrow[d, "B"] & q_2 \arrow[r, "A"] \arrow[d, "B"] & q_3 \arrow[r, "A"] \arrow[d, "B"] & \cdots \arrow[r, "A"] & q_T \arrow[d, "B"] \\
\text{Observation Sequence (O):} & O_1 & O_2 & O_3 & \cdots & O_T
\end{tikzcd}$$

Figure 4. Hidden Markov Model.

HMM predictors are very powerful adaptive stochastic models. Our hypothesis is that HMMs could compensate lack of relevant information through its underlying stochastic process that is not observable. A superior order HMM consists of the following elements [11]:

R – the order of HMM (a combination of *R* primitive hidden states form a so called super-state).

N – the number of primitive hidden states (belonging to a HMM of order 1), with $S = \{S_1, S_2, ..., S_{N^R}\}$ being the set of hidden super-states and $q_t \in S$ the hidden super-state at time *t*. The current super-state determines the transition into the next one based on a super-state transition matrix with restrictions [11]. *N* will be varied in order to obtain the optimal value.

M – the number of observable states, with $V = \{V_1, V_2, ..., V_M\}$ the set of observable states (symbols), and O_t the observable state at time *t*.

A = $\{a_{ij}\}$ – the transition probabilities between the hidden super-states S_i and S_j, where $a_{ij} = P[q_{t+1} = S_j | q_t = S_i], \quad 1 \le i, j \le N^R$.

B = $\{b_j(k)\}$ – the probabilities of the observable states V_k, considering the current hidden super-state S_j, where $b_j(k) = P[O_t = V_k | q_t = S_j]$, $1 \le j \le N^R$, $1 \le k \le M$.

π = $\{\pi_i\}$ – the initial hidden super-state probabilities, where $\pi_i = P[q_1 = S_i]$, $1 \le i \le N^R$.

The following variables are also defined [11, 24]:

$\alpha_t(i) = P(O_1 O_2 ... O_t, q_t = S_i | \lambda)$ – the forward variable, representing the probability of the partial observation sequence until time *t*, and hidden state S_i at time *t*, given the model $\lambda = (A, B, \pi)$.

$\beta_t(i) = P(O_{t+1} O_{t+2} ... O_T | q_t = S_i, \lambda)$ – the backward variable, representing the probability of the partial observation sequence from $t+1$ to the end *T*, given the hidden state S_i at time *t* and the model $\lambda = (A, B, \pi)$.

$\xi_t(i, j) = P(q_t = S_i, q_{t+1} = S_j | O_1 O_2 ... O_T, \lambda)$ – the probability of being in hidden state S_i at time *t*, and in hidden state S_j at time *t*+1, given the model $\lambda = (A, B, \pi)$ and the observation sequence.

$\gamma_t(i) = P(q_t = S_i | O_1 O_2 ... O_T, \lambda)$ – the probability of being in hidden state S_i at time *t*, given the model $\lambda = (A, B, \pi)$ and the observation sequence.

H – the history (the number of observations used in the prediction process). In [24] and [27] the entire observation sequence is used in the prediction process (*H*=*T*), but in some practical applications the observation sequence increases continuously, therefore its limitation is necessary [11].

I – the maximum number of iterations in the adjustment process. Usually the adjustment process ends when the probability of the last *H* observations does not increase anymore, but for a faster adjustment, the number of iterations is limited, as in [11].

The HMM-based prediction algorithm consists in the following steps:

1) T=H (T is the length of the observation sequence);
2) c=0 (c is the number of current iteration, its maximum value is given by I);
3) Initialize $\lambda = (A, B, \pi)$;
4) Compute $\alpha_t(i),\ \beta_t(i),\ \xi_t(i,j),\ \gamma_t(i),\quad t = 1,...,T,\quad i = 0,...,N^R - 1,\quad j = 0,...,N^R - 1$;
5) Adjust the model $\lambda = (A, B, \pi)$;
6) c=c+1;
7) If $P(O|\lambda)$ increases and c<I, go to 4.);
8) At time *T*, the next observation symbol O_{T+1} is predicted, using the adjusted model $\overline{\lambda} = (\overline{A}, \overline{B}, \overline{\pi})$:
   8.1) choose the current hidden state S_i, by maximizing $\alpha_T(i)$, $i = 0,...,N^R - 1$;
   8.2) choose the next hidden state S_j, by maximizing $\overline{a_{ij}}$,

   $$j = (i \text{ mod } N^{R-1}) \cdot N, ..., (i \text{ mod } N^{R-1}) \cdot N + N - 1;$$

   8.3) predict the next symbol V_k, by maximizing $\overline{b_j(k)}$, $k = 0,...,M - 1$.

Before a prediction, the model $\lambda = (A, B, \pi)$ is randomly initialized, as shown in [11], and after that it is repeatedly adjusted based on the last *H* observations $O_{T-H+1}, O_{T-H+2}, ..., O_T$ (the entire observation sequence if *H=T*), in order to increase the probability of the observation sequence $P(O_{T-H+1}\ O_{T-H+2}\ ...\ O_T|\lambda)$. In this work we use the Baum-Welch iterative algorithm introduced in [1] – identical with the Expectation Maximization (EM) method for this particular problem – which improves iteratively an initial model. The adjusted model $\overline{\lambda} = (\overline{A}, \overline{B}, \overline{\pi})$ is then used to predict the next observation symbol O_{T+1}.

In the multi-page prediction approach we prefetch the links whose probability is higher than 1/M. The predictor can be enhanced with confidence mechanism (like that from Figure 2).

3.3 Graph-Based Predictors

The graphs are data structures used in many types of applications to model different processes. A graph consists in vertices which can be connected by edges. We denote a graph G=(V, E), where V is the set of vertices and E is the set of edges. We also denote a path P={v$_1$, v$_2$, …, v$_k$}, where (v$_i$, v$_{i+1}$) is an edge, $i = \overline{1, k-1}$.

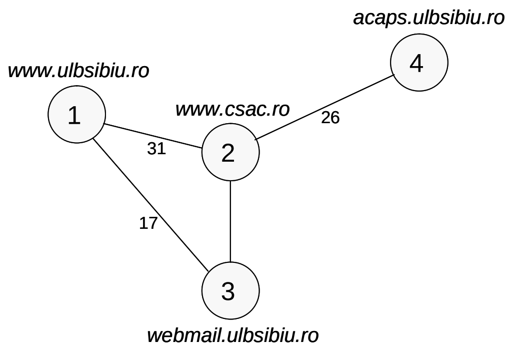

Figure 5. Modeling web page accesses using graphs.

As Figure 5 depicts, in our application we use undirected weighted graphs whose vertices represent the accessed links, an edge between two vertices meaning at least one transition from a link to another, whereas the weights are keeping the number of transitions.

Three graph algorithms have been analyzed from the web page prediction perspective. The weight-based and Dijkstra algorithms are statistical centric whereas the common-neighbors-based algorithm is topological centric. These algorithms can be enhanced with confidence mechanism (see Figure 2).

The weight-based algorithm predicts the next link as being the vertex whose edge with the current vertex has the highest weight. In the multi-page prediction approach all the adjacent vertices are prefetched. The multi-page prediction algorithm, enhanced with the confidence mechanism, is presented below:

```
c = current vertex (link)
if c.isPredictable() then
    predict c.getPairs()
update()
```

where *getPairs* returns a list with all the adjacent vertices and *update* adapts the confidence of the current link (by incrementing it on correct prediction or decrementing it on misprediction) and adds the new link to the graph.

The Dijkstra-based algorithm determines the highest path (having the highest sum of weights) and predicts the vertices from that path. This path is determined on the grounds of the well-known Dijkstra algorithm, presented in [5], which finds the paths with the lowest costs between a given source vertex and the other vertices from a weighted graph. Since we are interested in the highest path, we apply the Dijkstra algorithm with inversed weights (w^{-1}). In the multi-page prediction approach all the vertices from that path are prefetched. The multi-page prediction algorithm, enhanced with the confidence mechanism, is presented below:

```
c = current vertex (link)
pmax = Dijkstra(c)
if c.isPredictable() then
    predict pmax.getVertices()
update()
```

where *getVertices* returns a list with all the vertices from a path and *update* adapts the confidence of the current link and adds the new link to the graph.

The common-neighbors-based algorithm [21] discovers the vertex having the highest number of common neighbors with the current vertex and predicts these common neighbors. The multi-page prediction algorithm, enhanced with confidence mechanism, is given by the following pseudocode:

```
c = current vertex (link)
max = -1
foreach vertex v do
    neighbors = c.commonNeighbors(v)
    if neighbors.getSize() > max then
        max = neighbors.getSize()
        maxneighbors = neighbors
if c.isPredictable() then
    predict maxneighbors.getVertices()
update()
```

where *commonNeighbors* returns the list of vertices which are common neighbors for two given vertices, *getVertices* returns all the vertices from a list and *update* adapts the confidence of the current link and adds the new link to the graph.

3.4 Hybrid Predictors

Since we apply multi-page prediction, it is natural a hybrid prediction scheme which returns the predictions of all the components. All these predicted web pages are prefetched. In the next section we evaluate the predictors presented in the previous sections and we use the most accurate predictors as hybrid prediction components.

4 Experimental Results

The first step of our research is to comparatively analyze the proposed algorithms, from the prediction accuracy point of view, by varying their input parameters. The prediction accuracies presented in this section have been obtained through multi-page prediction. Such a study can be better highlighted on a set of log files. Therefore, the above presented algorithms have been implemented in Java and evaluated on two datasets.

The BU benchmark set was generated by the "Ocean Group Research" from Boston University [6] and consists in log files collected during 7 months on 37 workstations, spanning the timeframe from 21 November 1994 to 8 May 1995. Each log file name contains a user ID number, the machine on which the session took place and the Unix timestamp when the session started. Each line in a log corresponds to a single URL requested by the user; it contains the machine name, the timestamp when the request was made, the URL, the size of the document (including the overhead of the protocol) and the object retrieval time in seconds.

The evaluations have been performed also on the newer LBUS dataset collected from the educational web server of "Lucian Blaga" University of Sibiu, during 31 October 2006 – 10 November 2006. The log file contains about 210,000 web accesses, each line corresponding to an URL requested by the user.

First, we have evaluated the Markov predictor on the above mentioned datasets, by varying the pattern size (R). We used a history of 1000 links and 4-state confidence counters. The results are presented in Figure 6, where con112 - con99 are the BU log files.

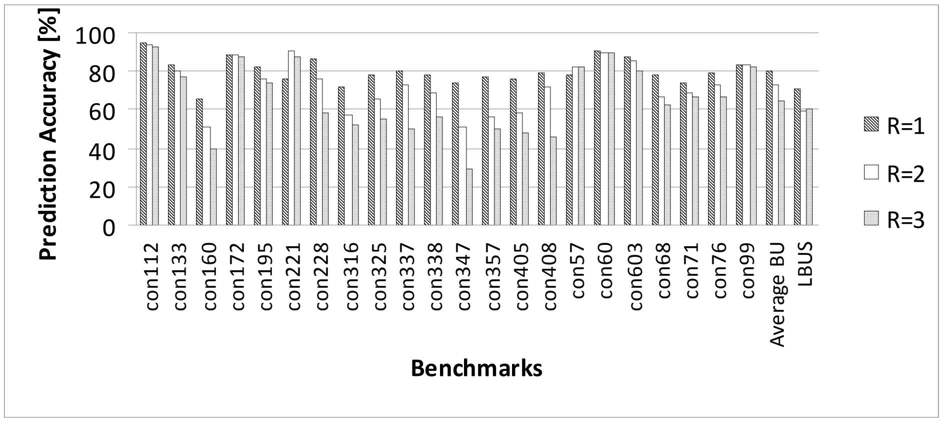

Figure 6. Web page prediction using Markov chains (H = 1000, 4-state confidence).

As Figure 6 shows, the first order Markov chain (R=1) was the most efficient on the evaluated data sets, which is in concordance with [20]. However, on some benchmarks (con172, con221, con57, con60, con603 and con99) superior order Markov chains performed better. The average prediction accuracy with a first order Markov chain was 80.07% on the BU dataset and 70.58% on LBUS.

Then, we have evaluated the HMM predictor on the same datasets, by varying the number of hidden states (N). We used a first order HMM with a history of 1000 links and 4-state confidence counters. The experimental results are presented in Figure 7.

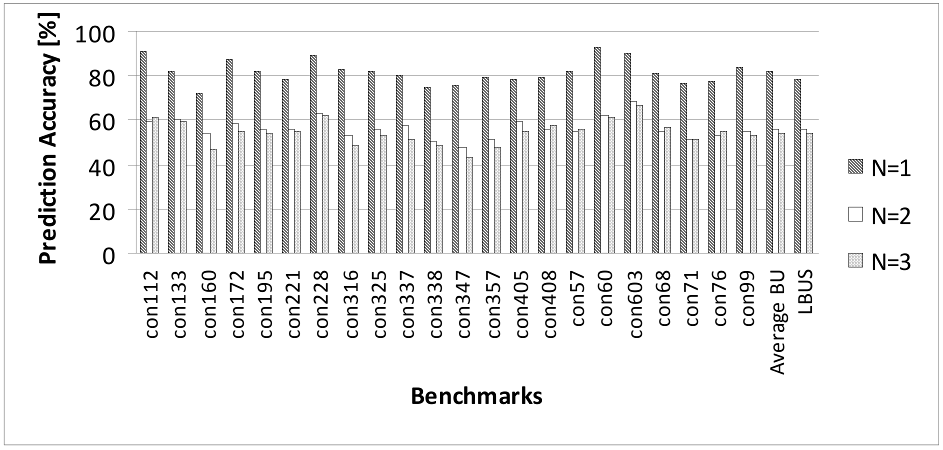

Figure 7. Web page prediction using HMMs (H=1000, R=1, 4-state confidence).

As the results show, the first order HMM with one hidden state (N=1) performed better on all benchmarks. The average prediction accuracy was 81.77% on BU and 78.29% on the LBUS dataset. We have evaluated also superior order HMMs (R=2, 3), obviously with higher number of hidden states, but the prediction accuracy was lower, which is consistent with what is reported in [20].

Finally, we have evaluated the graph based predictors. Figure 8 shows comparatively the prediction accuracies obtained with the weight-based predictor, the Dijkstra-based one and the common-neighbors-based (topological) one, using a history of 1000 links and 4-state confidence counters.

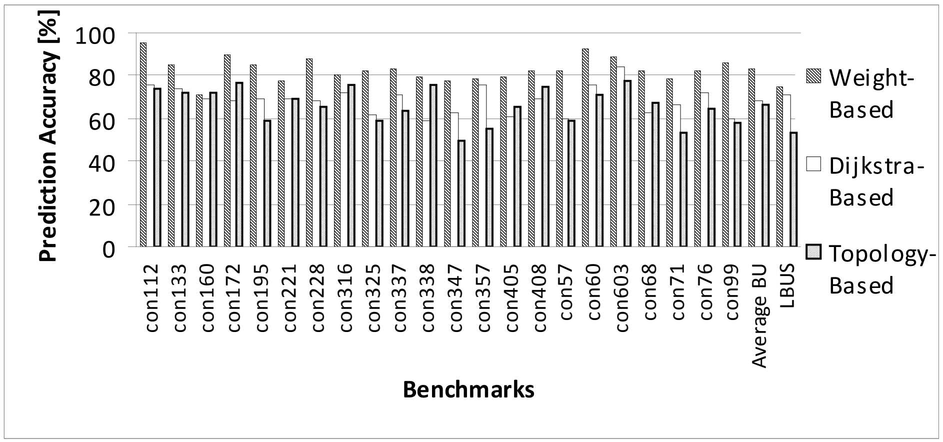

Figure 8. Web page prediction using graphs (H=1000, 4-state confidence).

As it can be observed, the weight-based predictor outperforms both the Dijkstra-based and topology-based algorithms on all the evaluated benchmarks. The average prediction accuracy, obtained with the weight-based graph predictor, was 83.07% on the BU dataset (but exceeding 90% on some benchmarks) and 75.06% on LBUS.

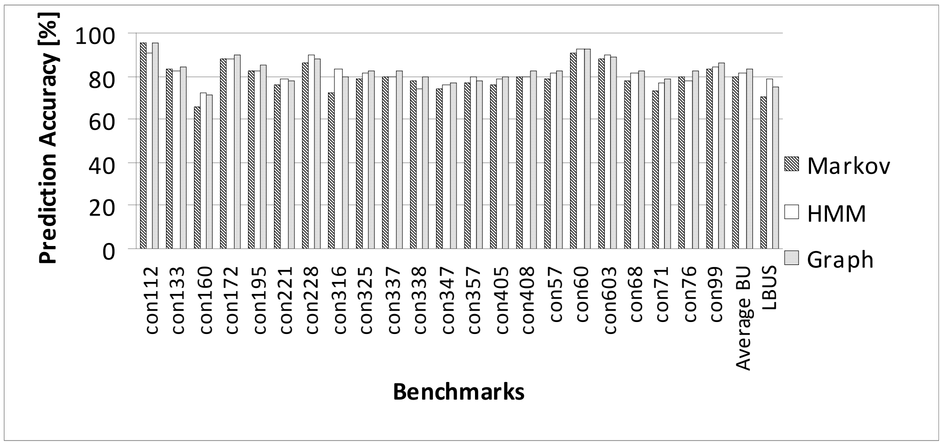

Figure 9. Comparing web page predictors (H=1000, 4-state confidence).

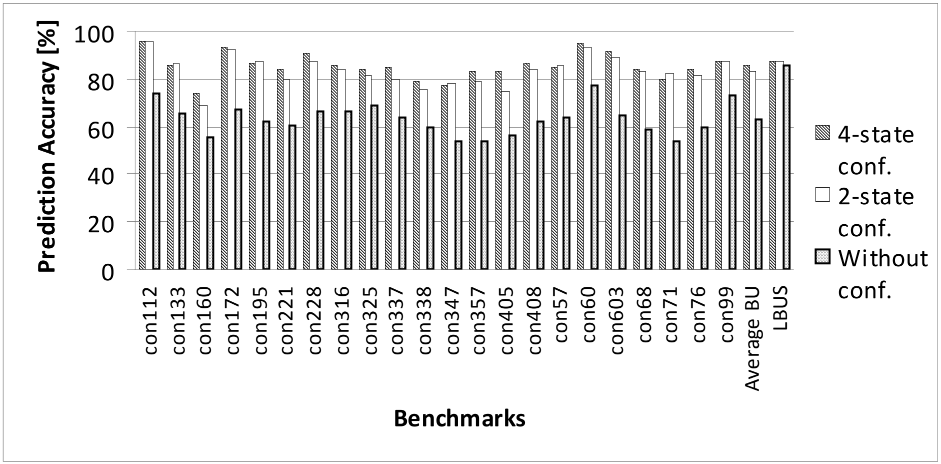

Figure 10. Comparing hybrid web page predictors with 4-state confidence, 2-state confidence and without confidence.

Figure 9 presents comparatively the best configurations of our evaluated predictors, all of them using the same history of 1000 links and 4-state confidence counters. The results show that on the BU dataset the weight-based graph predictor outperforms the other predictors, however, on some benchmarks (con160, con221, con228, con316, con357, con60 and con603) the HMM predictor showed the better result. On the LBUS dataset the HMM predictor was better. These results suggest that a hybrid predictor, consisting in a HMM and a graph-based predictor, could provide better results. Figure 10 shows comparatively the accuracies obtained using such hybrid predictors with 4-state confidence, 2-state confidence and without confidence, respectively. The hybrid predictor with 4-state confidence outperforms, in prediction accuracy, all the evaluated component predictors. Its average accuracy was 85.45% on BU and 87.28% on the LBUS dataset. It slightly outperforms the 2-state confidence based hybrid predictor and significantly outperforms the hybrid predictor which is not using any confidence mechanism. As Figure 10 shows, the confidence mechanism can increase the accuracy by avoiding prediction when the confidence in a certain context is low.

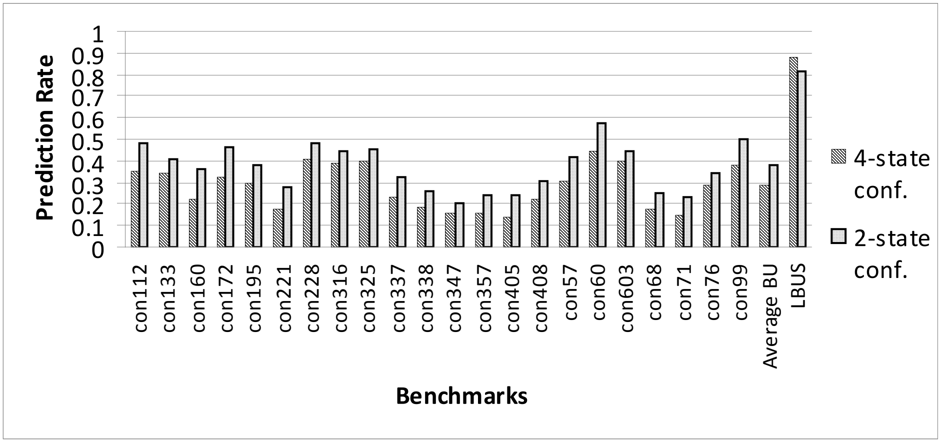

Figure 11. The prediction rates of the hybrid web page predictors with 4-state confidence and 2-state confidence.

Figure 11 depicts how the prediction rate (the number of predicted web pages divided to the total number of web pages) is influenced by the selectivity of the 4-state and 2-state confidence mechanisms in the case of the hybrid predictor. The prediction rate of the predictor without confidence is 1.0 because a prediction is always performed. It can be observed that as more selective confidence mechanism we have used as higher accuracy and lower prediction rate we obtained. An exception from this rule is only the LBUS log file where the prediction rate is lower with 2-state confidence than with 4-state confidence, at almost the same accuracy. For this dataset the faster transitions from predictable to unpredictable states, due to the lower number of predictable states in the 2-state confidence, is unfavorable. Thus, on the LBUS dataset a 4-state confidence mechanism is better from both the accuracy and prediction rate viewpoints.

The best predictors have been also integrated into the Mozilla Firefox browser as a JavaScript extension and have been evaluated in a real environment with good results.

Summary

In this section we have chosen the best Markov predictor (Figure 6), the best HMM predictor (Figure 7) and the best graph-based predictor (Figure 8), in terms of prediction accuracy. A comparison between these optimal predictors has been presented in Figure 9, which shows that the weight-based graph predictor outperforms the other evaluated predictors, but on some benchmarks, including the LBUS dataset, the HMM-based predictor has proved better. Finally, we have presented in Figure 10 the results obtained with a hybrid predictor having as components the HMM and the weight-based graph predictor and using different confidence mechanisms. Figures 10 and 11 also depict the benefit of the confidence mechanism.

However, the hybrid predictor with 4-state confidence provided the best prediction accuracy: 85.45% at average on BU and 87.28% on the LBUS dataset.

5 Conclusions and Further Work

In this study, we have presented three web page prediction methods in a comparative manner. We have included into our analysis Markov chains, Hidden Markov Models and graph algorithms. We have enhanced all these predictors with a confidence mechanism for higher prediction accuracy. The goal of the proposed predictive prefetching methods is to reduce the delays necessary to load the web pages and files visited by the users. The experimental results performed on the LBUS dataset have shown that the HMM predictor provided the best prediction accuracy, 78.29%. On the BU dataset the weight-based graph predictor provided the best prediction accuracy, 83.07% at average, but on some BU benchmarks the HMM predictor was better than the graph-based one, which suggests that a hybrid predictor, consisting in a HMM and a graph-based predictor, could provide better results. With such a hybrid predictor we obtained the best accuracy, 85.45% on BU and 87.28% on LBUS, being thus the most appropriate to predict and prefetch web pages efficiently. The evaluations have also shown that the first order Markov chain outperforms the superior order ones, as well as the first order HMM outperforms the superior order HMMs. Thus, since superior order predictors imply higher complexity but lower accuracy in both cases, in our opinion, more simple predictors are more suitable for web page prefetching. This is verified also in the case of graphs because the simpler weight-based predictor is more efficient than the Dijkstra-based one whose complexity is considerably higher.

The confidence mechanism has a high benefit since it can significantly increase the accuracy by avoiding predictions from low-confidence contexts.

A further work direction is the analysis of using neural networks and also other graph algorithms for web page prediction. Another further work direction consists in developing and evaluating different hybrid predictors. Another one could be the latency evaluation because it would be useful to show how retrieval times are reduced through prefetching. Finding and exploiting similarity among users is a research challenge in itself.

Acknowledgements

We express our gratitude to our MSc student Ionut-Madalin Ghimis for providing his useful and competent help in implementing the graph algorithms presented in this work.

References

1. Baum, L.E. An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of Markov Processes. Inequalities, Vol. 3, 1-8, 1972.
2. Cambazoglu, B.B. Algorithmic Techniques for Reducing Energy Consumption of Commercial Web Search engines. Yahoo! Research Barcelona, Spain, Training School, University of Balearic Islands, Palma de Mallorca, 2012.
3. Canali, C., Colajanni, M., Lancellotti, R. Adaptive algorithms for efficient content management in social network services. 10th International Conference on Computer and Information Technology, 68-75, 2010.
4. Ciobanu, D., Dinuca, C.E. Predicting the next page that will be visited by a web surfer using Page Rank algorithm. International Journal of Computers and Communications, Issue 1, Vol. 6, 60-67, 2012.
5. Cormen, T., Leiserson, C., Rivest, R., Stein, C. Introduction to Algorithms. MIT Press, Third Edition, 2009.
6. Cunha, C. A., Bestavros, A., Crovella, M. E. Characteristics of WWW Client Traces. Boston University Department of Computer Science, Technical Report TR-95-010, 1995.
7. Deshpande, M., Karypis, G. Selective Markov Models for Predicting Web-Page Accesses. ACM Transactions on Internet Technology, Vol. 4, Issue 2, 163-184, 2004.
8. Domènech, J., Pont, A., Sahuquillo, J., Gil, J. A. An experimental framework for testing web prefetching techniques. The 30th EUROMICRO Conference, 214-221, 2004.
9. Domènech, J., Sahuquillo, J., Pont, A., Gil, J. A. How current web generation affects prediction algorithms performance. Proceedings of SoftCOM International Conference on Software, Telecommunications and Computer Networks, 2005.
10. Dubey, S., Mishra, N. Web Page Prediction using Hybrid Model. International Journal on Computer Science & Engineering, Vol. 3 Issue 5, 2170-2176, 2011.
11. Gellert, A., Vintan, L. Person Movement Prediction Using Hidden Markov Models. Studies in Informatics and Control, Vol. 15, No. 1, National Institute for Research and Development in Informatics, Bucharest, 17-30, 2006.
12. Guo, Y. Z., Ramamohanarao, K., Park, L. A. F. Web Page Prediction Based on Conditional Random Fields. The 18th European Conference on Artificial Intelligence, 251-255, 2008.
13. Hasan, M. A., Chaoji, V., Salem, S., Zaki, M. Link Prediction using Supervised Learning. Proceedings of SDM 06 Workshop on Link Analysis, Counterterrorism and Security, 2006.
14. Huang, Q, Yang, Q., Huang, J. Z., Ng, M. K. Mining of Web-Page Visiting Patterns with Continuous-Time Markov Models. Springer-Verlag Berlin Heidelberg, 549-558, 2004.
15. Huang, Z. Link Prediction Based on Graph Topology: The Predictive Value of Generalized Clustering Coefficient. Proceedings of the Workshop on Link Analysis: Dynamics and Static of Large Networks, 2006.

16. Jin, X., Xu, H. An Approach to Intelligent Web Pre-fetching Based on Hidden Markov Model. Proceedings of the 42nd Conference on Decision and Control, Maui, Hawaii, USA, 2954-2958, Vol. 3, 2003.
17. Kaushal, P. Hybrid Markov model for better prediction of web page. International Journal of Scientific and Research Publications, Vol. 2, Issue 8, 2012.
18. Khalil, F., Li, J., Wang, H. Integrating Recommendation Models for Improved Web Page Prediction Accuracy. Proceedings of the 31st Australasian Conference on Computer Science, Vol. 74, 91-100, 2008.
19. Khalil, F., Li, J., Wang, H. An Integrated Model for Next Page Access Prediction. Internation Journal of Knowledge and Web Intelligence, Vol. 1, No. 1/2, 48-80, 2009.
20. Khanchana, R., Punithavalli, M. Web Page Prediction for Web Personalization: A Review. Global Journal of Computer Science and Technology, Vol. 11, Issue 7, 39-44, 2011.
21. Liben-Nowell, D., Kleinberg, J. The Link-Prediction Problem for Social Networks, Journal of the American Society for Information Science and Technology, Vol. 58, Issue 7, pages 1019-1031, 2007.
22. Murata, T., Moriyasu, S. Link Prediction of Social Networks Based on Weighted Proximity Measures. IEEE/WIC/ACM International Conference on Web Intelligence, 85-88, 2007.
23. Petzold, J. Augsburg Indoor Location Tracking Benchmarks. Technical Report 2004-9, Institute of Computer Science, University of Augsburg, Germany, 2004.
24. Rabiner, L.R. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE, Vol 77, No. 2, 257-286, 1989.
25. Schmidt, E. Every 2 days we create as much information as we did up to 2003. Lake Tahoe, CA. URL http://techcrunch.com/2010/08/04/schmidt-data/, 2010.
26. Singhai, N., Nigam R.K. A Novel Technique to Predict Oftenly Used Web Pages from Usage Patterns. International Journal of Emerging Trends & Technology in Computer Science, Vol. 1, Issue 4, 49-55. (2012)
27. Stamp, M. A Revealing Introduction to Hidden Markov Models. http://www.cs.sjsu.edu/faculty/stamp/RUA/HMM.pdf, 2004.
28. Su, Z., Yang, Q., Zhang, H. J. A Prediction System for Multimedia Pre-fetching in Internet. Proceedings of the eighth ACM international conference on Multimedia, 3-11, 2000.
29. Temgire, S., Gupta. P. Review on Web Prefetching Techniques. International Journal of Technology Enhancements and Emerging Engineering Research, Vol. 1, Issue 4, 100-105, 2013.
30. Vintan, L., Gellert, A., Petzold, J., Ungerer, T. Person Movement Prediction Using Neural Networks. Proceedings of the KI2004 International Workshop on Modeling and Retrieval of Context (MRC 2004), Vol-114, Ulm, Germany, 2004.
31. Vintan, L., Florea, A., Gellert, A. Random Degrees of Unbiased Branches. Proceedings of the Romanian Academy, Series A, No. 3, 259-268, 2008.
32. Wan, M., Jönsson, A., Wang, C., Li, L., Yang, Y Web user clustering and Web prefetching using Random Indexing with weight functions. Knowledge and Information Systems, Vol. 33, Issue 1, 89-115, 2012.
33. Zhu, J., Hong, J., Hughes, J. G. Using Markov Chains for Link Prediction in Adaptive Web Sites. Springer-Verlag Berlin Heidelberg, 60-73, 2002.