Predicting Web User's Next Access Based on Log Data

Rituparna Sen
University of Chicago

Mark H. Hansen
Bell Laboratories

Abstract

In this article we consider models that describe how people browse the Web. We restrict our attention to navigation patterns within a single site, and base our study on standard Web server access logs. Given a visitor's previous activities on the site, we propose models that predict their next page request. If the prediction is reasonably accurate, we might consider "prefetching" the page before the visitor requests it. A more conservative use for such predictions would be to simply update the freshness records in a proxy or network cache, eliminating unnecessary `If-Modified-Since` requests. Using data from the Web site for the Computing and Mathematical Sciences Research Division of Lucent Technologies (`cm.bell-labs.com`) we first evaluate the predictive performance of low-order Markov models. We next consider mixtures of first-order Markov models, achieving a kind of clustering of Web pages in the site. This approach is shown to perform well, while significantly reducing the space required to store the model. Finally, we explore a Bayesian approach using a Dirichlet prior on the collection of links available to a user at each stage in their travels through the site. We show that the posterior probabilities derived under this model are fairly close to the cross-validation estimates of the probability of success.

1 Introduction

The Web is a big place. The search engine `www.google.com`, for example, claims to have indexed over a billion separate Web pages. Given the large number of resources on the Web, sites are constantly competing to attract and retain new visitors. To improve a user's experience, it is now common for sites to customize content (generating pages based on a user's previous activities) and to consider creative caching and prefetching schemes to deliver content as quickly as possible. In both cases, a model for describing how visitors navigate a Web site can be invaluable.

In this paper, we derive a predictive model for the pages a user will visit, given some portion of their previous browsing history. We restrict our attention to the path a user follows within a single Web site. Our study is based on data from `cm.bell-labs.com`, a site for the Computing and Mathematical Sciences Research Division of Bell Laboratories, Lucent Technologies. Suppose that while visiting this site, a user requests pages $x_0, \ldots, x_t$ in order. Our goal is to predict what will happen next. Will the user leave, deciding to visit another site or pursue another activity? If the user remains within the domain `cm.bell-labs.com`, what page will they request next? To address this problem, we investigate several probability models of the form

$$P(x_{t+1}|x_t, x_{t-1}, \ldots, x_0)\,. \tag{1}$$

The state space consists of all the pages contained in the entire Web site and one additional symbol to denote that the user left. Simple Markov models are a popular choice for P and are at the heart of Web usage analysis [2] and prefetching schemes (Eric Grosse, personal communication, July, 2000). We have found that the predictions from this model are rather poor and in this paper we examine several alternatives.

Section 2 describes the raw data for our study and outlines how they are processed prior to modeling and analysis. Section 3 presents several classes of models, and in Section 4 a decision-theoretic framework is proposed to compare their performance. In Section 5, we comment on ongoing work to extend these models.

2 Web Logs

As you navigate the Web, your browser requests various kinds of data from one or more Web servers. As part of the process of delivering content, most Web servers will record information about the visitor and the items they requested. These items include HTML pages, images, Java class files and PostScript documents. The information available to the Web server about each request includes a timestamp, the IP address of the

Table 1: Log entries for single request to `cm.bell-labs.com/stat/`.

Client IP	*Date*	*Time*	*Item*	*Bytes*	*Code*	*Referral*
cs.washington.edu	8/10/2000	15:23:04	/stat/index.html	9269	200	http://search.yahoo.com/search?p=network+stats
cs.washington.edu	8/10/2000	15:23:04	/stat/style/win.css	486	200	http://cm.bell-labs.com/stat/index.html
cs.washington.edu	8/10/2000	15:23:04	/stat/images/logo.gif	6562	200	http://cm.bell-labs.com/stat/index.html
cs.washington.edu	8/10/2000	15:23:05	/stat/images/tukey.gif	4441	200	http://cm.bell-labs.com/stat/index.html
cs.washington.edu	8/10/2000	15:23:05	/stat/images/pbar.gif	145	200	http://cm.bell-labs.com/stat/index.html

visitor's computer, the type of browser they are using, the name of the requested item, and the "referral" (the address of the page that directed the visitor to the requested item). These details are typically stored as a single line of a (potentially enormous) log file. Sample entries in this file are given in Table 1. These lines were generated by a request for the page `cm.bell-labs.com/stat/index.html`. The user was referred to this page by a Yahoo search for `network+stats`. Accessing the HTML file `index.html` spawned requests for a style sheet and three images (the next four lines in Table 1).

On a typical day, the visits to `cm.bell-labs.com` result in a 4Mb (compressed) log file, consisting of 1.5M entries. By Internet standards, this is small. A site like `www.yahoo.com` can easily compile 3Gb of log data per day. Given that our interest is in how users navigate the content of a Web site, we restrict our attention to HTML files, PostScript and PDF documents. All other requests recorded in the log, such as images, style sheets, and Java class files, are ignored. This reduces the data by a factor of 10. Because each request is logged as it occurs, the log file is organized in time order. To be useful for modeling, we have to process the reduced file to form a data set consisting of user paths or "visits". (In the middle of the day, there can be as many as 100 visitors on this Web site, each simultaneously generating lines in the log.) Formally, a visit is a sequence of content requests made by a single user while browsing the site. The processing steps from raw log file to usable visit data are given below:

Data reduction: Eliminate requests for items other than HTML files, PostScript and PDF documents.

Grouping into visits: In this step, we essentially replay the reduced log file, examining each request in time order. During this process, we identify separate visitors using the IP address of their computer. This simple approach can lead to difficulties if many visitors browse the site at the same time from the same computer or via the same proxy with ip-masquerading. Using a simple filter, we scanned the logs for this effect and found only a handful of such instances. A more accurate technique for identifying separate visitors requires either setting cookies or forcing visitors to register when they arrive at the site. We then build up the visit, compiling all of the user's content requests. When we decide that a user has left the site we store the visit in a separate data structure. The rules used to end a visit depend on the amount of time that has elapsed since the user's last request. In addition, we examine whether or not the page they are requesting is linked to one of the pages they previously viewed (including the referral page, when available). For almost all the visits, a simple rule that ends a session when the user has been inactive for 15 minutes is sufficient.

Table 2: Visit summaries for training and test data.

		Summary statistics of length of visits				
Sample	*No. Visits*	*Mean*	*Var*	*95%*	*99%*	*Max*
Learning	539401	3.0	680.5	9	21	11562
Test	53421	3.0	146.1	9	22	1322

Data cleaning: We next examine the visits for anomalies. In particular, we attempt to identify and remove the requests generated by robots. A robot is an automated program that skims the Web usually to populate the index of a search engine. Not surprisingly, a robot requests pages in a manner that is different from a human user. Often the number of pages requested is unbelievably large (sometimes numbering in the thousands over a five minute span) or the pattern of requests is spread across very different parts of the site. Typically, a robot will identify itself by requesting a file called `robots.txt`. Using this criterion, we systematically remove all visitors with IP addresses that are known to be robots. Unfortunately, because many robots do not access `robots.txt`, this strategy does not completely eliminate the problem. To catch the remaining robots, we have had to form various "predictability" rules for removing the others. We will mention these in Section 3.3.

To evaluate the performance of each of the models introduced in the next section, we divide the visit data into training and test sets: all visits from 04/05/2000 to 07/28/2000 are used for model fitting, while those from 07/31/2000 to 08/10/2000 are used as a test sample. In Table 2 we present summary statistics of visit lengths for the two samples.

3 Navigation models

Although the reduction to "content" requests like HTML and PostScript files allows us to focus on a smaller data set, the state space for the navigation model (1) is still huge. While `cm.bell-labs.com` is a small site, it consists of nearly 35,000 content items. This fact is important when constructing a predictive model that will have practical value. We begin by considering low-order Markov models. While we find that a second order description performs reasonably well, storage constraints force us to develop a space-efficient approximation. Therefore, we next derive a mixture of first-order models that also fits the data but requires far fewer system resources. Finally, we study an entirely different kind of approach that uses the link structure on the requested pages. We propose a Bayesian model that makes use of this information and demonstrate that it also has favorable predictive power.

3.1 Markov Models of Order 1 and 2

Huberman et al. [3] propose a Markov model for browsing behavior conditional on the length of a visit, L. They assign L a two parameter inverse Gaussian distribution,

$$P(L) = \sqrt{\frac{\lambda}{2\pi L^3}} \exp\left[\frac{-\lambda(L-\mu)^2}{2\mu^2 L}\right] . \tag{2}$$

The lengths of visits to `cm.bell-labs.com` are plotted in Figure 1a. For the complete data set, the maximum likelihood estimates are $\hat{\mu} = 2.994$ and $\hat{\lambda} = 0.184$. According to [3], a log-log plot of length versus frequency should roughly follow a straight line with a slope of -1.5 for small values of L. This holds for our data as seen in Figure 1a. In Figure 1b, we present an empirical CDF of visit length together with the estimated CDF based on (2). Huberman et al. [3] offer plots like this in support of their model. However, in Figure 1c, a P-P plot of the same data uncovers rather serious misfit. A closer examination of the plots in [3] suggests that these authors probably had similar difficulties with the inverse Gaussian model.

Given a value of the length variable L, Huberman et al. [3] model a sequence of requests using a simple first order Markov chain. As users will typically select their next request from the collection of links on the page they are viewing, some kind of Markov structure is not unbelievable. Rather than take the approach of [3], we construct Markov models where the state space includes an "exit" symbol. We begin with a first order model, and estimate the transition matrix $P(x_{t+1}|x_t)$ from the complete set of visit data. We then use this model to simulate visits. In Figure 2a, we present a Q-Q plot to compare the visit lengths of the generated data with the visits seen in the training data. The maximum length of the simulated visits is 47 while that of the observed data is above 1200. The generated data also contain many cycles (multiple requests for the same sequence of pages). As a result, the proportion of unique pages in a visit tends to be much smaller in the simulations.

To overcome these deficiencies, we next consider a second order Markov model. The transition matrix is again estimated from the observed visits. In Figure 2b, we present a Q-Q plot comparing simulated and

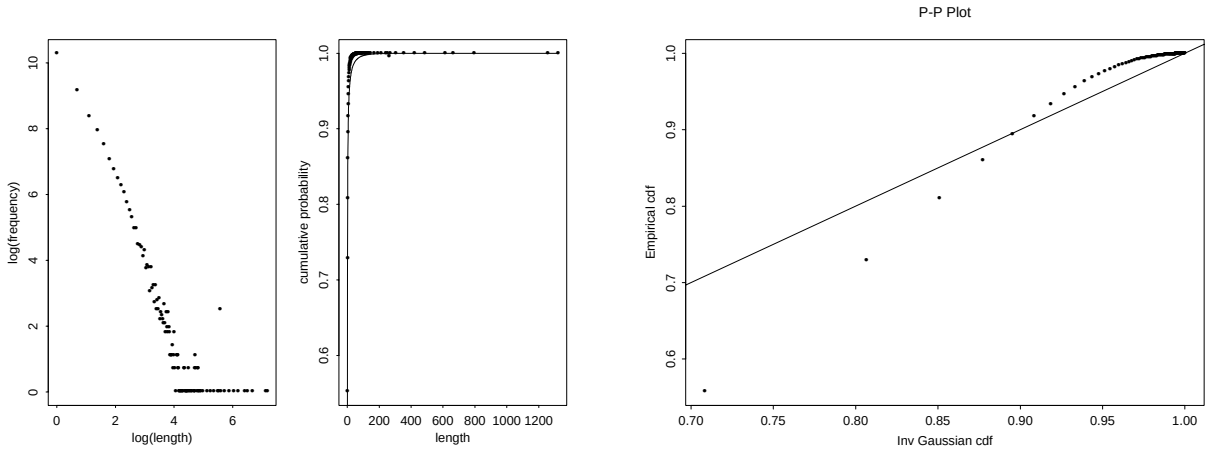


Figure 1: (a) Plot of log(frequency) on log(length); (b) Inverse Gaussian CDF with empirical CDF; (c) P-P plot of Inverse Gaussian fit against empirical data.

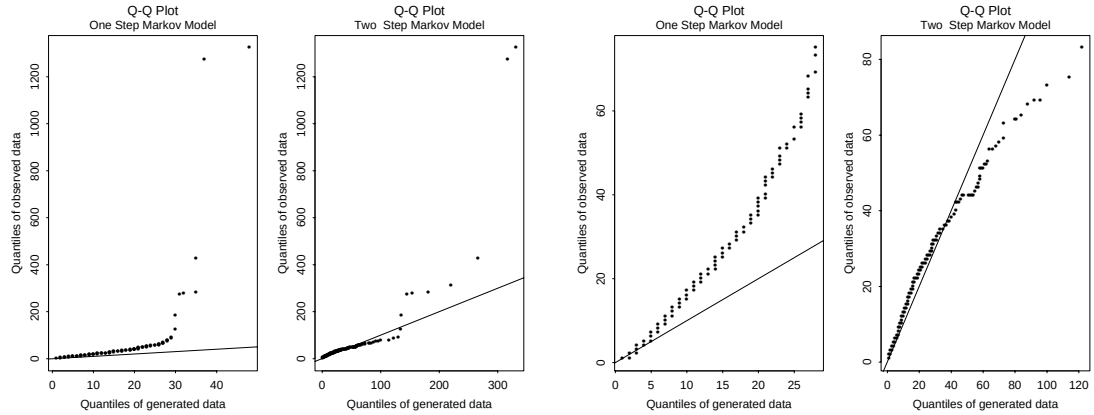


Figure 2: Q-Q plots of visit lengths for Markov models of order (a) One; and (b) Two. (c) & (d) are cropped versions of (a) & (b)

observed visit lengths. Figure 2c and 2d are cropped versions of Figure 2a and 2b, leaving out less than 0.1% of the data in each case. These plots are more effective at illustrating the quality of the fit. While we have greatly reduced the misfit for long sessions, the second order model still tends to produce shorter visits than those seen in the log file. An anonymous referee suggested a mixture model that fit separate transition matrices for long and short visits. As an alternative, we believe that some of this misfit can be reduced with so-called variable length Markov models. We discuss this approach briefly in Section 5 as it is the focus of our ongoing research. To further explore the second order model, in Figure 3 we present a series of P-P plots comparing the distribution of the number of unique pages in a visit conditional on length of visit. Despite some concerns about the length of a small number of very long sessions, these plots and other analysis lead us to believe that the second order model is adequate for a simple site like `cm.bell-labs.com`.

3.2 Mixture Model

Although the second order Markov model works well, the number of parameters involved is enormous. This can degrade the accuracy of the estimated transition probabilities as well as complicate computation and storage. As an intermediate approach, we consider finite mixtures of first order models. The structure is similar in spirit to the Mixture Transition Distribution Model introduced by Raftery [5]. Formally, we consider grouping the pages into classes. We study the performance of models with 2, 3, 4 and 5 classes.

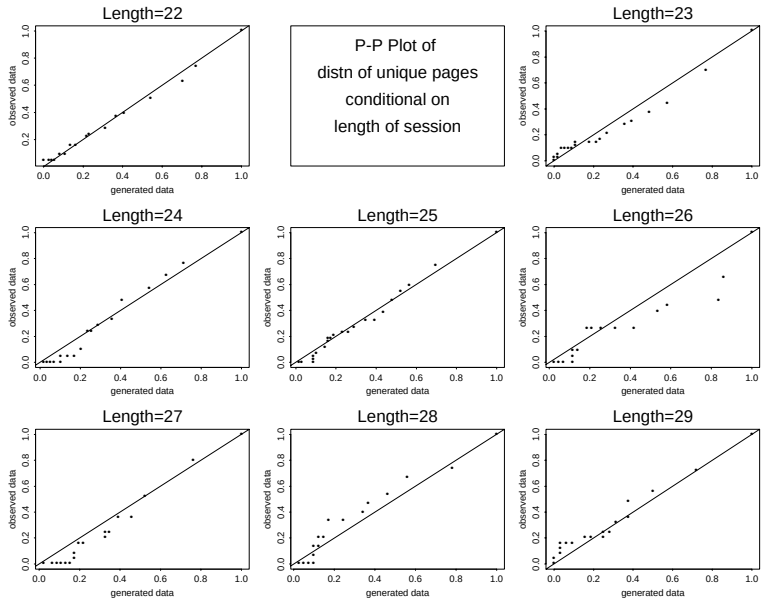


Figure 3: P-P plots comparing the number of unique pages in a visit, observed versus generated data.

The classes are created to maximize the likelihood of the model we now specify.

Let C denote the set of classes and let π_{iI} be the probability that page i belongs to class $I \in C$. Then, we write

$$P(x_{t+1} = k | x_t = j, x_{t-1} = i) = \sum_{I \in C} \pi_{iI} P_I(k|j)\,, \tag{3}$$

implying the mixture is with respect to the class of the request at $t-1$. Take n_{ijk} to be the number of transitions in our training data for which $x_{t+1} = k$, $x_t = j$ and $x_{t-1} = i$. As is common when fitting mixture models, we apply the EM algorithm [4]. This requires the introduction of latent variables z_{iI} which equal one if page i belongs to class I and zero otherwise. Then the so-called "complete data" log likelihood is

$$\sum_{i \in W} \sum_{j \in W} \sum_{k \in W} n_{ijk} \sum_{I \in C} z_{iI} \log\left(\, \pi_{iI} P_I(k|j) \,\right)\,,$$

and the two phases of the EM algorithm are given by

$$\text{The M-step:}\quad \hat{P}_I(k|j) = \frac{\sum_{i \in W} n_{ijk} \hat{\pi}_{iI}}{\sum_{i \in W} n_{ij\cdot} \hat{\pi}_{iI}} \qquad \text{The E-step:}\quad \hat{\pi}_{iI} = \frac{\prod_{j \in W} \prod_{k \in W} P_I^{n_{ijk}}(k|j) \hat{\pi}_{iI}}{\sum_{J \in C} \prod_{j \in W} \prod_{k \in W} P_J^{n_{ijk}}(k|j) \hat{\pi}_{iJ}}$$

The EM algorithm was carried out by initially allocating each page randomly to a class. Figure 4 shows a plot of the difference between $\hat{p}_{ijk}$ obtained by direct estimation of the second order Markov model and the mixture scheme using equation (3). The difference is displayed in a series of boxplots, where the horizontal axis is a function of the support (n_{ijk}). It is seen that for large support the two are almost same. This is expected. But for small values of n_{ijk} it seems the mixture estimates are in general smaller.

3.3 A Bayesian Approach

As a visitor proceeds through different pages in a Web site, it seems sensible that their next request x_{t+1} should come from a set S of all clickable links contained on any of the previous pages $x_0, \ldots, x_t$. Note that the set S is enlarging with each request and depends on the visit. In our data, the majority of visitors made requests almost exclusively from this set. For example, we observed that for 88% of the visits at least 2 of the first 4 moves were from S. Through manual inspection, we found that the IP addresses of the remaining

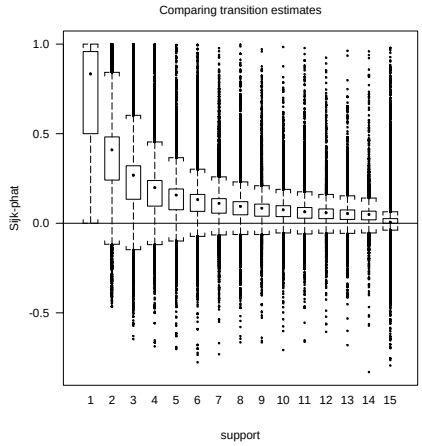


Figure 4: Differences in estimated transition probabilities p_{ijk} for second order and the mixture models.

12% corresponded to search engines and hence the requests were probably made by robots. This idea has been developed into a criterion for recognizing robots.

As a first attempt to use link information for modeling navigation patterns, we construct a prior distribution on the transition matrix. For simplicity, we let S_i denote the collection of m_i content links on page i, and let p_{ij} denote the probability $P(x_{t+1} = j | x_t = i)$. From the visit data, we extract the number of transitions n_{ij} from i to j and set $\sum_{j \in W} n_{ij} = n_{i.}$, where W denotes the state space of all the pages on the site and an extra symbol to represent the end of the visit. Assuming a multinomial likelihood for p_{ij}, we assign a conjugate Dirichlet prior with parameters $(\alpha + 1)$ for pages in S_i and 1 for all other $(N - m_i)$ pages in the site:

$$P(p_{ij}) = \frac{\Gamma(m_i \times \alpha + N) \prod_{j \in S_i} p_{ij}^{\alpha}}{\{\Gamma(\alpha + 1)\}^{m_i}}, \qquad \text{where} \quad \sum_{j \in S_i} p_{ij} = 1 .$$

Let $\mathcal{D}$ be the transition data $\{n_{ij}\}$, so that we can write the posterior for p_{ij} as

$$P(p_{ij} | \mathcal{D}) = \frac{\Gamma(m_i \times \alpha + n_{i.} + N) \prod_{j \in S_i} p_{ij}^{\alpha + n_{ij}} \prod_{k \notin S_i} p_{ik}^{n_{ik}}}{\prod_{j \in S_i} \{\Gamma(\alpha + n_{ij} + 1)\} \prod_{k \notin S_i} \{\Gamma(n_{ik} + 1)\}}, \quad \text{where} \quad \sum_{j \in S_i} p_{ij} + \sum_{k \notin S_i} p_{ik} = 1 .$$

This distribution achieves its maximum at

$$\hat{p}_{ij} = \frac{\alpha I(j \in S_i) + n_{ij}}{m_i \alpha + n_{i.}} = \frac{I(j \in S_i)}{m_i} \times \frac{m_i \alpha}{m_i \alpha + n_{i.}} + \frac{n_{ij}}{n_{i.}} \times \frac{n_{i.}}{m_i \alpha + n_{i.}}$$

We use these posterior values to predict a visitor's next request. This approach gives similar results for a wide range of α values. The success probability is not very different from either cross-validation or test-set estimates. Therefore, because we can compute the Bayesian quantities directly, this approach is well suited for real-time prediction schemes. So far, we have only described the case where we use links from page x_t to help predict x_{t+1}. The Bayesian scheme studied in Section 4 incorporates link information from the complete history $x_0, \ldots, x_t$. A detailed treatment of this approach is beyond the scope of this paper. In the next section, we consider the predictive performance of all the schemes introduced so far.

4 Performance

In addition to pure predictive ability, we also evaluate these models in terms of their performance in a prefetching scheme. To reduce the amount of time waiting for Web pages, your browser or perhaps a local

Table 3: Probability of Inclusion

Method	Size of list			
	1	2	3	20
2nd Ord Markov	0.28	0.45	0.52	0.75
2 Clusters	0.24	0.38	0.47	0.80
3 Clusters	0.25	0.39	0.47	0.79
4 Clusters	0.26	0.390	0.508	0.78
5 Clusters	0.25	0.402	0.510	0.78
Bayes $\alpha = 20$	0.17	0.24	0.29	0.76
Bayes $\alpha = 40$	0.17	0.23	0.29	0.76

Table 4: Analysis of loss for prefetching

Pred.	Obs.	Case	Prob	Loss w/ pre	Loss w/o pre
file1	file1	3	P_3	f	$f+d$
out	file2	2	P_2	$f+d$	$f+d$
no pred	out	1	P_1	0	0
out	out	0	P_0	0	0
file1	out	-1	P_{-1}	f	0
no pred	file2	-2	P_{-2}	$f+d$	$f+d$
file1	file2	-3	P_{-3}	$2f+d$	$f+d$

cache can arrange to have available the items you are likely to visit next. For example, your browser can begin downloading a number of most likely next pages based on their size, the probability of being requested and current network traffic. If you request one of these items, they will be immediately ready for viewing. Obviously, it is best to keep the number of items downloaded in this fashion to a minimum, requesting only those with very high probability of being required later. A more conservative approach uses a predictive model to update cache freshness records for your most likely next requests, eliminating unnecessary `GET If-Modified-Since` messages. This is used on the server `cm.bell-labs.com` where 96% of the `If-Modified-Since` queries get a `Not-Modified` response. This scheme can make use of longer lists of predicted next moves without the risk of generating unnecessary traffic. The freshness approach was originally implemented using a first order Markov model(Eric Grosse, personal communication, July, 2000).

For all the models considered, the parameters are estimated based on the learning sample and predictive performance is studied on the test sample described in Table 2. The results are summarized in Table 3. The probability of inclusion is the proportion of times the page that was ultimately requested was included in the list of predictions. A list of length m, for example, consists of the m "next pages" having greatest probability under the given model. From this table, it is clear that the second order Markov model performs better if we are interested in actually prefetching 1, 2, or 3 pages. The mixture model with two classes performs best if the implementation is in a proxy-server checking for cache freshness and we are allowed bigger lists. In this application, the second order model requires too much storage space, so a small number of first order transitions is preferred.

In Table 4, we list several possible outputs from a predictive model: we can suggest one or more files as being the next request; we can predict that the visitor will leave the site; or we can decide that we have insufficient information and not make a prediction. On the other hand, the visitor can request the predicted file, or another file, or leave. The histogram in Figure 5a shows the performance of the different methods in these various cases. But it is very difficult to compare the different methods based on this histogram since there are so many possibilities. So we apply a decision theoretic approach to determine a criterion for comparison. Let f denote the cost of prefetching a page and let d be the cost of a dissatisfied visitor who has to wait while a page downloads. Then the loss for all the possibilities in the two circumstances of whether prefetching is done or not are summarized in Table 4. Notice that the only difference in loss is in cases 3, -1 and -3.

$$\text{Risk if prefetching} - \text{Risk if not prefetching} = -d \times P_3 + f \times (P_{-3} + P_{-1}).$$

Therefore the method that minimizes this is the best. The relevant probabilities to compare are P_3 and $P_{-3} + P_{-1}$. The exact comparison will of course depend on f and d. A barplot of P_3 and $P_{-1} + P_{-3}$ for different models is shown in Figure 5b. In both Figure 5a and 5b the last bar (marked "Large") shows the predictive performance of the second order Markov model for visits of length greater than 5. The prediction is best in this case. This shows that users with long visits are more predictable. The second order Markov model (labeled "M2" in Figures 5ab) tested on all visits is the next best. This is consistent with Table 3 as we are predicting lists of size 1.

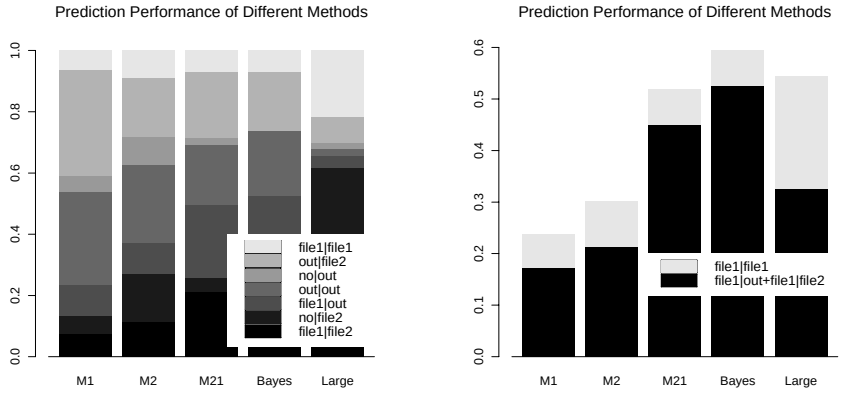


Figure 5: Performance of each method in a prefetching scheme.(a) All Possibilities; (b)Relevant Probabilities from decision theoretic approach. M1 denotes a first order Markov model tested on visits of all lengths; M2 is a second order Markov model also tested on visits of all lengths; M12 is the first-order mixture approximation to M2, tested using visits of all lengths; Bayes is the Bayesian model based on the link structure of the pages viewed by a user during thier visit, also tested with visits of all lengths; and Large is the second order Markov model tested only on visits of length 5 or more. In (a) for the Bayes column, the combinations "no — out" and "no — file2" are essentially zero and do not appear.

5 Future Work

Most of the frequent sequences observed in the data are of very short length. However, significant fraction are quite long. For example, there are 2010 sequences of requests of length 15, each repeated more than 4 times (keep in mind that the average length of a visit to this site is only 3). Common, repeated sequences of requests occur mainly when a user is going through a list of clickable links provided on a page. More modest repetitions occur because of pages containing multiple frames. Because of this skewed distribution of observed sequences, a variable length Markov model [1] seems very appropriate for the data. We are currently investigating this model and early results are promising.

6 References

[1] Buhlmann, P., and Wyner, A. (1999) "Variable length Markov chains," *Ann Statist*, 27, 480-513.

[2] Cadez, I., Heckerman, D., Meek, C., Smyth, P., and White, S. (2000), " Application of mixtures of Markov models to clustering and visualization of several hundred thousand page request sequences from a large commercial Web site," *Proceedings KDD2000*.

[3] Huberman, B. A., Pirolli, P. L. T., Pitkow, J. E., and Lukose, R. M. (1998) "Strong Regularities in World Wide Web Surfing," *Science*, 280, 95-97.

[4] McLachlan, G. J., and Krishnan, T. (1996) *The EM Algorithm and Extensions*, New York, NY: John Wiley.

[5] Raftery, A. E. (1985) "A Model for High-Order Markov Chains," *Journal of the Royal Statistical Society*, Ser. B, 47(3), 528-539.