Predicting Users' Next Access to the Web Server

by

Oxana Chakoula

B.S., Moscow State University, 1994

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES

(Department of Computer Science)

We accept this thesis as conforming
to the required standard

The University of British Columbia

January 2003

Department of Computer Science

The University of British Columbia
Vancouver, Canada

Date 16 Jan 2003

Abstract

The World Wide Web is experiencing a rapid growth both in the volume of traffic and complexity of web sites. Two of the main directions of research on improving users' web browsing experience are reduction of network latency perceived by users, and website personalization. Both tasks require the development of models that can predict a user's next request to a web server.

Markov models trained on web logs have been found well suited for addressing this problem. Most of the variations of Markov models suggested for predicting a user's next request to a web server base their predictions on the data from the entire user population. Our hypothesis is that clustering users' web sessions to reflect browsing patterns of particular groups of users will improve prediction accuracy.

We consider two clustering techniques that use different representations of web sessions. One approach treats web sessions as Markov models of order zero, and the other represents them as Markov models of order one. We compare the two clustered prediction models to the state-of-the-art non-clustered Markov models of orders 0, 1, All-2 and All-3. We report empirical results based on web logs of UBC Computer Science department.

We found clustering web sessions as Markov chains performed better than clustering them of Markov models of order zero, yet the former did not improve predictability over the non-clustered first order Markov model.

Contents

List of Tables

List of Figures

Acknowledgements

I am very grateful to my supervisor, David Poole, for his guidance, ideas, discussions, comments, and financial support. Special thanks go to my second reader, Nando de Freitas, for his valuable advice and ideas. I thank my husband, Serguey, and my daughter, Anastasiya, for their love, patience, and encouragement.

OXANA CHAKOULA

The University of British Columbia
January 2003

Chapter 1

Introduction

Websites are created with their users in mind. Their content, link structure, types of interactions available, and "look-and-feel" reflect their webmasters' view on how a site is intended to be used. Usually, webmasters do not receive much of direct feedback from users. Logs collected by web servers do provide feedback, albeit indirect.

Web servers collect huge amounts of data every day. The server of the UBC Computer Science department web site, `www.cs.ubc.ca`, gathers about 50MB log entries daily. Near one tenth of those are requests for HTML files, while the rest are requests for auxiliary files.

Extracting the trails users actually follow and comparing them to intended site usage can help with justification and optimization of a site link structure. Namely, it can suggest which links should be added, or how the information should be restructured.

Being able to predict the next file requested from a web server makes it possible to pre-send this file to the client's computer. If predictions are correct often enough, the network latency perceived by users can be significantly reduced.

Being able to elicit users' browsing patterns and preferences can help facilitate and personalize users' web experience in terms of adaptive web pages and agent

assisted navigation.

It has been observed that users tend to repeat the trails they have followed once [18]. So, better prediction of a user's next request could be made on the data pertaining to that particular user, not all the users. However, this would require reliable user identification and tracking users between sessions. This is usually achieved by sending cookies to a client browser, or by registering users. Both require user cooperation and might discourage some of potential site visitors. So many websites choose not to use these means of user tracking. Also, building prediction models on individual data would reqiure that users have accessed enough pages to make a prediction, which is not usually the case for a university website that has many casual users.

We hypothesize that the population of users can be divided into groups, each of them having their distinct browsing patterns. And, without storing information about particular users, we can still benefit from the diversity of their interests.

We concern ourselves with the task of learning users' web browsing behaviour from web logs, and utilizing this knowledge for improving web site usability. Particularly, we consider the problem of predicting users' accesses to a Web server.

Contributions

Most of the approaches suggested for predicting a user's next request to a web server base their predictions on the data from the entire user population.

In this thesis we consider clustering user sessions prior to learning a prediction model. First, we cluster web sessions as Markov models of order zero. Second, we cluster them as discrete Markov chains. We compare the two clustered prediction models with the state-of-the-art All-K^{th} Markov model. We report empirical results based on web logs of UBC Computer Science department.

Outline of the document

The rest of the thesis is organized as follows. The related previous work is presented in Chapter 2. In Chapter 3 we outline our approach. We report our results in Chapter 4. In Chapter 5 we conclude and discuss future work.

Chapter 2

Literature Review

The problem of eliciting users' browsing patterns from web logs has been investigated for different purposes and in a variety of ways.

Next request prediction can be used for latency reduction as well as for user characterization. We discuss this in Section 2.1.

Finding groups of similarly minded users based on their browsing behaviour is used for user characterization and creating adaptive web pages. It can benefit prediction as well. We discuss clustering in Section 2.2.

Web logs contain very raw data. Prepocessing is needed to transform a log into a set of user web sessions suitable for analysis. Web log preprocessing is a task separate from prediction. Yet, the assumptions made and the options chosen at this stage effect the performance and applicability of prediction models. Section 2.3 is devoted to this issue.

2.1 Markov Models and Next Request Prediction

A web session is a sequence of pages traversed by a user during their single visit to a site. Web sessions can be extracted from a web log (Section 2.3).

Markov models [16] have been widely used for modeling and predicting stochastic processes. Particularly, they proved to be an adequate and robust repre-

sentation of web browsing behaviour [5], [27].

A Markov model of order K is a set of three parameters $< \mathbf{S}, \mathbf{C}, \mathbf{T} >$. $\mathbf{S}$ is a set of all possible states for which the model is built. $\mathbf{C}$ is a set of all contexts of length K. A *context* is a sequence of states that have been observed in a given order, where there was a next state. $\mathbf{T}$ is a $|\mathbf{C}|$x$|\mathbf{S}|$ transition probability matrix, where each entry T_{ij} corresponds to the probability of state j occurring next given that the context i has been observed.

The first order Markov model (Markov chain) predicts the next state by only looking at the last state. Here the set of contexts $\mathbf{C}$ is a subset of $\mathbf{S}$. The zeroth order model does not look at history and always predicts the mode state. For this model the transition probability matrix is a 1x$|\mathbf{S}|$ vector of state occurrence probabilities.

In next web request prediction context, states correspond to all web pages that can be requested by users.

For prediction, a web log is usually divided into two parts, training set and test set. The training set is used to learn a set of contexts and a transition matrix. The test set is used to evaluate predictive performance of a model.

For each sequence of states of length K in the test set, where there is a next state, the prediction is made. If the corresponding sequence of previous states is present in the set of contexts, the *prediction* is a state having maximal transition probability. The prediction is *correct* if the predicted next state is the same as the actual one, and incorrect otherwise. If there is no corresponding sequence of states in the set of contexts, the prediction is *null*, which is treated as an incorrect prediction.

An alternative setting of the problem implies making multiple predictions. Instead of a single prediction, a prediction set of a predefined maximal size is generated. In this setting a prediction is *correct* if the prediction set is non-empty and the actual next request belongs to the set, and incorrect otherwise.

A Markov model can be characterized by its *predictive accuracy* $p(Hit)$ [27]

defined as the number of correct predictions ("hits") divided by the total number of states to be predicted. The *coverage* $p(Match)$ of a model is the percentage of cases when the model is able to make a prediction, i.e. for a context encountered in the test set there is the matching entry in the set of contexts. The two are connected via *conditional predictive accuracy*, which is the probability of a hit if there is a match: $p(Hit) = p(Match)p(Hit|Match)$. The model *complexity* is the number of rows in its transition probability matrix.

The model size grows exponentially with the model order [9], [19], [26], [27]. For the single prediction setting, higher order models typically have higher conditional predictive accuracy but lower coverage [9], [19], [26], [27]. Their product $p(Hit)$ shows no unique trend as the model order increases: it was higher for the second order model than for the first order model as reported by Su et al. [26], Zukerman et al. [27], and Deshpande and Karypis [9], but it was lower in experiments by Pitkow and Pirolli [19].

Bestavros [5] employed a first order Markov model to implement a server-side page prefetching service. This considerably reduced server load and service time.

Sarukkai [22] compared the performance of the first order Markov model for prediction of next user activity and next request to the server. They found that predictive accuracy of both models monotonically increases as the size of prediction set increases, and that the performance of next user activity prediction is better than that of server request prediction.

All of the Markov models discussed in this thesis except the following one use only page(s) requested previously to predict the page requested next. Zukerman et al. [27] modified this setting by using additional information from web log, namely the referrer of a current page (Section 2.3), instead of the page requested last. The two are the same if pages are traversed sequentially without backtracking. Otherwise browser or proxy caching (Section 2.3) causes them to be different. The modification helps partially overcome session distortion introduced by caching by

taking into account the tree structure of the site. Zukerman et al. [27] use the name "space" for the modified models, while the original ones are referred to as "time models" as they are based on time ordered sequences of pages.

Four variations of first and second order Markov models were considered by Zukerman et al. [27]:

1. First order Time Markov model. The prediction is based on the document requested last.

2. Second-order Time Markov model. The prediction is based on the document requested last and the document requested before that.

3. Space Markov model. The prediction is based on the referrer of the requested document.

4. Linked Space-Time Markov model. The prediction is based on the document requested last and its referrer.

The Linked model performs the best [27] as it takes into account both the order in which documents are requested and the structure of the site.

The All-K^{th} order Markov model [19] combines Markov models of all orders from 1 to K. During the prediction phase the models are tried in sequence from the highest order to the lowest. The prediction is taken from the highest order model that is able to make one, i.e. has a corresponding matching sequence of pages in its set of contexts.

Palpanas [15] built a prefetching engine using All-K^{th} order Markov model, and tested its predictive accuracy, usefulness of predictions, and latency reduction effectiveness as functions of model order, prediction set size, session time threshold, and client cache size.

The All-K^{th} model typically gives better prediction than any single order model [19], [26], as it combines high coverage of lower order models with better

predicting power of higher order models. However, the problem of model complexity is exacerbated further.

With its huge number of states the All-K^{th} order Markov model is highly redundant, and some of its states can be ignored without significantly affecting predictive performance. A number of ways has been developed to intelligently select subsets of higher order Markov models in order to reduce model complexity yet retain its high predictive power.

Schechter et al. [23] and Pitkow and Pirolli [19] constructed their prediction models by including all the paths of order one (i.e. single pages), and only those higher order paths that occurred more than once. During prediction step, longer matching paths were preferred to shorter ones; the most frequent path was chosen among those of equal length.

This approach, called Longest Repeating Subsequence (LRS) in [19], is equal to All-infinity Markov model, except that only the sequences that occured repeatedly (i.e. at least twice) contribute to the model.

Pitkow and Pirolli [19] compared the LRS approach to All-K^{th} order Markov models for K from 1 to 7. They found that the LRS model has significantly lower complexity accompanied by only a slight decrease in predictive power.

In this approach, an implicit frequency threshold of two is used to determine which sequences to include in the model.

Deshpande and Karypis [9] generalized the LRS model by allowing for an arbitrary frequency threshold, and suggested two more techniques for model pruning. These techniques were also based on thresholds for confidence and number of errors respectively. The models compared were:

1. Support-Pruned Markov model. An LRS model with a variable frequency threshold.

2. Confidence-Pruned Markov model. A sequence of states is retained only if the probability of the most frequent next state is significantly different from the

probabilities of the other next states.

3. Error-Pruned Markov model. Training data are divided into two parts. The first part is used to build the model; the second part (validation set) is used to prune it.

The authors reported that all three models achieved even better prediction accuracy than the initial All-K^{th} order model. From the first model to the last, the model complexity increases, and so does predictive accuracy.

This may seem to contradict with the results of [19]. But different research groups used different datasets and different preprocessing options. We discuss preprocessing in Section 2.3.

All prediction methods discussed so far are based on data for the entire user population, without dividing users into groups.

2.2 Web Session Clustering

Clustering web sessions can be used for web site personalization. Namely, sets of links that could be of interest to a certain group of users are extracted to form new index pages. This index page can be just added to the site, or it can be shown to the corresponding group of users as their personalized version of the main page.

Perkovitz and Etzioni [17] built clusters of related but not linked web pages. Their clustering is based on page co-occurrence frequencies in users' session. An order in which pages are accessed is not taken into account.

Kamdar and Joshi [14] clustered the user sessions based on pair-wise dissimilarities between the sessions. Similarity between two URLs is measured as an overlap in the paths from the root of the web site tree to the corresponding nodes.

Banerjee and Ghosh [2] clustered web sessions for user profiling. Their similarity measure between any two paths is based on their longest common subsequence

(LCS). For each path, a sequence of times spent on the pages in the LCS is extracted. These two time vectors are aggregated into the similarity measure.

All these authors used variations of the K-means algorithm [12]. This is a popular and robust algorithm for clustering. However, K-means algorithm requires specifying a distance measure, which is not always intuitive.

Expectation-Maximization (EM) is a probabilistic approach that does not require specifying a distance measure.

Ridgeway [21] and de Freitas [8] developed the Expectation-Maximization method for clustering Markov chains. Cadez et al. [6] used the EM clustering of web sessions for user browsing behaviour visualization. However, they did not apply clustering directly to web pages. Instead, they divided web pages into a small number of groups, and considered user sessions as sequences of transitions between these groups.

Sen and Hansen [24] applied the EM to web log clustering for prediction. They found that for the larger prediction sets, clustered first order model outperformed the second order model. Yet they did not study this systematically with respect to model order and the size of prediction set.

2.3 Web Log Preprocessing

HTTP/1.0 [4], the original version of HTTP, is a stateless protocol. A client computer establishes a separate connection to a server for every file requested. The later version of the protocol, HTTP/1.1 [10], allows a client to make multiple requests using the same connection. As both protocols are widely used, web log formats reflect the semantics of the former protocol.

Table 2.1 presents a fragment of web log in Extended Web Log Format [1], [11]. These are log entries generated by user requesting a single page `www.cs.ubc.ca/index.html`. The first entry is the request for an HTML file, the rest are requests for auxiliary files embedded in the web page.

IP address	Date and Time	Method	Item	Protocol	Status	Bytes	Referrer
x.x.x.x	31/May/2002:00:32:01	GET	/index.html	HTTP/1.1	200	9224	-
x.x.x.x	31/May/2002:00:32:02	GET	/style.css	HTTP/1.1	200	439	http://www.cs.ubc.ca/
x.x.x.x	31/May/2002:00:32:02	GET	/images/index-03.gif	HTTP/1.1	200	9224	http://www.cs.ubc.ca/
x.x.x.x	31/May/2002:00:32:02	GET	/images/Random10.jpg	HTTP/1.1	200	17456	http://www.cs.ubc.ca/
x.x.x.x	31/May/2002:00:32:02	GET	/images/announcements-2.gif	HTTP/1.1	200	898	http://www.cs.ubc.ca/
x.x.x.x	31/May/2002:00:32:02	GET	/images/UBC-Logo-down.gif	HTTP/1.1	200	2088	http://www.cs.ubc.ca/
x.x.x.x	31/May/2002:00:32:02	GET	/images/research-over.gif	HTTP/1.1	200	943	http://www.cs.ubc.ca/
x.x.x.x	31/May/2002:00:32:02	GET	/images/grad-over.gif	HTTP/1.1	200	953	http://www.cs.ubc.ca/
x.x.x.x	31/May/2002:00:32:02	GET	/images/people-over.gif	HTTP/1.1	200	663	http://www.cs.ubc.ca/
x.x.x.x	31/May/2002:00:32:02	GET	/images/employment-over.gif	HTTP/1.1	200	1116	http://www.cs.ubc.ca/

Table 2.1: Web log fragment

Thus web logs bear no notion of user sessions. To extract user sessions from a web log, several preprocessing steps are needed.

Common pre-processing tasks include [7]:

Cleaning out erroneous requests

The HTTP status code returned to the client [1], [11] indicates whether or not the file was successfully retrieved, and if not, what error message was returned. 2xx codes indicate successful retrieval, where 3xx, 4xx and 5xx codes indicate redirection, client side and server side errors respectively. Only 2xx log entries are retained for analysis.

Identifying pageviews

As in our example, several files usually contribute to a single pageview. Identifying pageviews includes cleaning out auxiliary files of certain types (e.g images, Java classes, cascade style sheets etc.).

Identifying users

In a web log, an IP address or domain name of a client computer is recorded. In the absence of cookies or user registration, this is the only means of user identification.

Usually an IP address corresponds to a single user.

This is not the case, however, if a proxy server is used to service a community of users. If the community is large enough chances are that more than one user from the same IP address will browse the same site simultaneously, so the web log would erroneously record these overlapping activities as one session.

A symmetric case that violates the one-to-one assumption is a session composed of temporarily overlapping activities inside one site initiated by a single user from several computers with different IP-addresses. Yet the evidence suggests [3] that this does not happen often.

For many web log mining tasks, it is important to identify web browsing sessions made by *human* users, as opposed to those made by robots. Robots are automated agents that browse broadly, quickly, and in an order different from humans. Robots are required to request the file `robots.txt` upon their entry to a site, so "well-behaved" robots can be detected using this rule. If a site does not have such a file, requests for it results in "404 – file not found" response. These can be detected but only before cleaning logs of erroneous requests.

Identifying sessions

Berendt et al. [3] evaluated the accuracy of several commonly used heuristics for dividing web logs into sessions. They reconstructed sessions from logs in three ways and compared the reconstructions with actual user activities they monitored. The sessionizing heuristics they compared were:

1. A session is a sequence of pages traversed inside a 30-minute time limit.
2. A session is a sequence of pages each of which was viewed less than 10 minutes.
3. A session is a sequence of pages where each page (except the first one) is reachable from one of the previously requested pages.

They found that the first two heuristics reconstructed true sessions well, while the last one often failed to do so.

Inferring cached requests

If a client browser has its cache enabled, after a (non-dynamic) page was requested once, its copy is placed in the cache, so all the successive requests to this page will bypass the server and will be serviced from the cache. Particularly, this happens when a user hits the Back button, which comprises about one third of all user's activities on the web.

Some researchers think that such missed log entries should be inferred and added to the log during preprocessing step [7], [9].

The others claim that it is not needed since:

- hitting the Back button is often not a meaningful user action in the sense that the user really wanted to see the previous page again. More probably, the user wanted to see its sibling page [25].

- often only pages viewed for the first time are of special interest. In many applications such as shopping cart analysis the expert is interested in the pages accessed during a session but not in the order of access [3].

- reconstruction of cached requests from the site tree may result in several possible paths between two pages, if the site is highly connected. Also, a user could have transferred to a page via History or Bookmark options, or by typing the URL in. In this case, *any* reconstruction of this non-existing user path would be incorrect.

Most preprocessing steps are heuristic in their nature. The set of preprocessing steps is influenced by web log mining goals. Usually researchers whose goal is to predict next requests to the server include less preprocessing options than those who predict next actions of users.

	Remove					*Add*
	Erroneous requests	Images	pages	Dynamic Proxies	Robots	Cached pages
Bestavros [5]	yes	no[1]				
Schechter et al. [23]			yes/no[2]			
Zukerman et al. [27]		yes				
Sarukkai [22]		yes/no[3]	yes			
Deshpande and Karypis [9]	yes	yes	yes	yes[4]	yes	yes
Sen and Hansen [24]	yes	yes	yes		yes	
Su et al. [26]		yes	yes			

Table 2.2: Preprocessing options

Table 2.2 summarizes the preprocessing options used by several research groups for different purposes. In the table yes entries denote options included in preprocessing, no denote options explicitly excluded, yes/no denote double experiments with an option both included and excluded. Blank entries refer to preprocessing steps not mentioned. Surprisingly, cleaning out erroneous requests is often among those.

Study of influence of preprocessing options on model predictive performance is beyond the scope of this work. Yet preprocessing options as well as different nature of web logs used are issues to be taken into account when comparing results of different web request prediction methods.

[1] Image files are treated in the same way as HTML files. Time threshold is used to control the degree of dependency between pages. Low time threshold causes only auxiliary pages to be dependent, higher threshold includes transition pages as well.

[2] One experiment was run with dynamic pages included. The other one treated dynamic pages as static, ignoring parameters.

[3] An experiment with images was done for server load prediction; an experiment without images was done for user activity prediction.

[4] Requests from proxy servers were detected and decomposed into individual user sessions.

Chapter 3

Prediction models

Our idea is to cluster sessions before learning prediction models. The goal is to obtain more specific models that reflect browsing patterns of particular groups of users, and to improve predictability in this way.

We compare two clustering techniques that use different representations of web sessions. One approach treats web sessions as Markov models of order zero (texts), and the other represents them as Markov models of order one (Markov chains).

Section 3.1 provides the data specification. Text clustering approach is presented in Section 3.2. Markov chain clustering is discussed in Section 3.3. Section 3.4 explains the prediction step common to both models.

3.1 Data

A web log is a set of web sessions divided into two subsets, the training set and the test set. A web session is a sequence of web pages. Let the training set contain n_s web pages, denoted by integers 1, 2, ..., n_s.

Let the training set consist of n_x sessions $\mathbf{x}_k = (x_{k1}, x_{k2}, \ldots, x_{kn_k})$, where x_{ki} belongs to the set $\{1, 2, \ldots, n_s\}$, and $k = 1, \ldots, n_x$.

Similarly, the test set consists of n_y sessions $\mathbf{y}_k = (y_{k1}, y_{k2}, \ldots, y_{kn_k})$, where

y_{ki} belongs to the set $\{0, 1, 2, \ldots, n_s\}$, and $k = 1, \ldots, n_y$. An additional state 0 is introduced to denote pages that have not been observed in the training set.

The training set is used to learn the models. The test set is used to evaluate their predictive performance.

3.2 Clustering web sessions as texts

We use the Probabilistic Latent Semantic Analysis model for the analysis of text by Hofmann [13].

The model represents texts as "bags of words", i.e. Markov models of zeroth order. Analogously, we treat web sessions as "bags of web pages". The key assumption is that this simplified representation will preserve most of the relevant information needed for prediction.

Each web session can be represented as a frequency vector $\mathbf{M}_k$, where element $M_{j,k}$ is the frequency of occurrence of page j in session k.

These frequency vectors can be modeled by a multinomial distribution. To cluster these sequences into n_c groups, we model these vectors as generated by a finite mixture of multinomial models:

$$\mathbf{M}_k|\boldsymbol{\theta} \overset{iid}{\sim} \sum_{c=1}^{n_c} p(c) \left[\prod_{j=1}^{n_s} p(j|c)^{M_{j,k}} \right] \tag{3.1}$$

where $\boldsymbol{\theta} \triangleq \{p(c), p(j|c)\}$ is the set of the model parameters: $p(c)$ is the probability of each cluster, $p(j|c)$ is the probability of state j in cluster c.

We assume that the parameters $p(c)$ and $p(j|c)$ are drawn from independent identical Dirichlet prior distributions with parameters α and β respectively.

We obtain the parameter estimates from the training set via Expectation-Maximization algorithm for maximum a posteriori estimation. The process iterates between the following two steps:

E step: computing the expected value of the complete log-likelihood function

using the set of parameters from the previous step. Essentially the step consists of computing cluster memberships of each session:

$$\widehat{p}(c|k) \propto \widehat{p}(c) \prod_{j=1}^{n_s} p(j|c)^{M_{j,k}} \tag{3.2}$$

and normalising it over all the clusters.

M step: finding the new values of parameters that maximize the expected value of the complete log-likelihood function computed in the E step.

$$\widehat{p}(c) = \frac{\alpha - 1 + \sum_{k=1}^{n_x} p(c|k)}{n_c(\alpha - 1) + n_x} \tag{3.3}$$

$$\widehat{p}(j|c) = \frac{\beta - 1 + \sum_{k=1}^{n_x} M_{j,k} p(c|k)}{n_s(\beta - 1) + \sum_{j=1}^{n_s} \sum_{k=1}^{n_x} M_{j,k} p(c|k)} \tag{3.4}$$

Having computed cluster membership of each session $p(c|k)$, we can calculate $\widehat{p}(j|i,c)$, probabilities of transition from state i to state j in cluster c. Here $N_{i,j,k}$ denotes the frequency of transition from page i to page j in session k.

$$\widehat{p}(j|i,c) = \frac{\sum_{k=1}^{n_x} N_{i,j,k} p(c|k)}{\sum_{j=1}^{n_s} \sum_{k=1}^{n_x} N_{i,j,k} p(c|k)} \tag{3.5}$$

3.3 Clustering web sessions as discrete Markov chains

We apply clustering first order Markov models (Markov chains), the approach developed by de Freitas [8].

Each web session can be represented as a transition table $\mathbf{N}_k$, where element $N_{i,j,k}$ is the frequency of transition from page i to page j in session k.

These transition tables can be modeled by a multinomial distribution.To cluster these sequences into n_c groups, we model these tables as generated by a finite mixture of multinomial models:

$$\mathbf{N}_k|\boldsymbol{\theta} \overset{iid}{\sim} \sum_{c=1}^{n_c} p(c) \left[\prod_{j=1}^{n_s} p(j|c)^{\mathbb{I}_j(x_{k1})} \prod_{j=1}^{n_s} \prod_{i=1}^{n_s} p(j|i,c)^{N_{i,j,k}} \right] \tag{3.6}$$

where $\boldsymbol{\theta} \triangleq \{p(c), p(j|c), p(j|i,c)\}$ is the set of the model parameters: $p(c)$ is the probability of each cluster, $p(j|c)$ is the prior probability of state j in the cluster c, and $p(j|i,c)$ is the transition probability from state i to state j in cluster c. $\mathbb{I}_j(x_{k1})$ is the initial state indicator: $\mathbb{I}_j(x_{k1}) = 1$ if $x_{k1} = j$ and 0 otherwise.

We assume that the parameters $p(c)$, $p(j|c)$, and $p(j|i,c)$ are drawn from independent identical Dirichlet prior distributions with parameters α, β, and γ respectively.

We obtain the parameter estimates from the training set via Expectation-Maximization algorithm for maximum a posteriori estimation.

E step: computing cluster memberships of each session:

$$\widehat{p}(c|k) \propto \widehat{p}(c) \left[\widehat{p}(x_{k1}|c) \prod_{j=1}^{n_s} \prod_{i=1}^{n_s} p(j|i,c)^{N_{i,j,k}} \right] \tag{3.7}$$

M step: finding the new values of parameters:

$$\widehat{p}(c) = \frac{\alpha - 1 + \sum_{k=1}^{n_x} p(c|k)}{n_c(\alpha - 1) + n_x} \tag{3.8}$$

$$\widehat{p}(j|c) = \frac{\beta - 1 + \sum_{k=1}^{n_x} \mathbb{I}_j(x_{k1}) p(c|k)}{n_s(\beta - 1) + \sum_{k=1}^{n_x} p(c|k)} \tag{3.9}$$

$$\widehat{p}(j|i,c) = \frac{\beta - 1 + \sum_{k=1}^{n_x} N_{i,j,k} p(c|k)}{n_s(\beta - 1) + \sum_{j=1}^{n_s} \sum_{k=1}^{n_x} N_{i,j,k} p(c|k)} \tag{3.10}$$

3.4 Prediction from clustered models

Let the test set consist of n_y sessions: $\mathbf{y}_k = (y_{k1}, y_{k2}, \ldots, y_{kn_k})$, where y_{ki} belongs to the set $\{0, 1, 2, \ldots, n_s\}$, and $k = 1, \ldots, n_y$.

In text clustering model, we have learnt $\widehat{p}(c)$ and $\widehat{p}(j|c)$ from the EM process (3.3 - 3.4) and computed $\widehat{p}(j|i,c)$ from session transition matrices (3.5). In Markov chain clustering model, we have learnt $\widehat{p}(c)$, $\widehat{p}(j|c)$, and $\widehat{p}(j|i,c)$ from the EM process (3.8 - 3.10).

For each beginning of a test session, $\mathbf{y}^\star = \{y_{k1}, y_{k2}, \ldots, y_{kt}\}$, we compute its membership to each cluster:

$$p(c|\mathbf{y}^\star) \propto p(\mathbf{y}^\star|c)p(c) \approx \widehat{p}(c) \left(\prod_{j=1}^{n_s} \widehat{p}(j|c)^{\mathbb{I}_j(y^\star_{k1})} \prod_{j=1}^{n_s} \prod_{i=1}^{n_s} \widehat{p}(j|i,c)^{N^\star_{i,j,k}} \right) \tag{3.11}$$

Then the probabilities of next step can be calculated as follows:

$$p(y^\star_{t+1} = j | y^\star_t = i) = \sum_{c=1}^{n_c} \widehat{p}(j|i,c)p(c|\mathbf{y}^\star) \tag{3.12}$$

The state $j = argmax(\widehat{p}(j|i))$ comprises a model prediction. If an experiment setting allows making multiple predictions (Section 2.1), the prediction set is the set of states j with top transition probabilities $\widehat{p}(j|i)$.

The predictive accuracy of a model is calculated as the number of correct predictions divided by the total number of states to be predicted (Section 2.1).

Chapter 4

Results

Our task is to compare predictive performance of text clustering to that of Markov chain clustering, and to compare them to non-clustered Markov models of orders 0, 1, All-2 and All-3 (section 2.1).

4.1 Data

We ran our experiments on UBC Computer Science department web log data.

In order to make a comparison across several datasets, we extracted four subsets from the logs: graduate, undergraduate, and prospective students' pages, and pages for the Computational Intelligence textbook [20]. We refer to the datasets as *Grad*, *Ugrad*, *Prospective*, and *Book* respectively.

Web server logs for January and February were used as training dataset, and logs for March were used as test dataset.

Our preprocessing included removing images and auxiliary files, and sessionizing according to the 10 minute per page heuristic [3].

We excluded sessions of length 1 both from modeling and prediction. Also, we restricted upper bound of session length by 100 by truncating the longer sessions.

We assumed that each IP-address corresponded to a different user.

The descriptive statistics of the datasets are presented in Table 4.1. The

Dataset		No. Sessions	No. Transitions	Session length Min	Max	Mean	Var
Book	Training	3724	23473	2	100	7.3	13.7
	Test	1723	16986	2	100	10.9	20.5
Grad	Training	4393	11718	2	100	3.7	4.0
	Test	1617	4776	2	100	4.0	5.9
Ugrad	Training	15237	54156	2	100	4.6	7.4
	Test	7552	25737	2	100	4.4	6.1
Prospective	Training	5096	14184	2	66	3.8	2.6
	Test	2282	5915	2	28	3.6	2.4

Table 4.1: Descriptive statistics of datasets

datasets show that the subsites have different usage patterns. For all but one dataset average session length is rather low. Lower mean can indicate that many of a subsite visitors are either users whose browsing goal is simple and well defined, or casual users whose goal pages do not belong to this subsite.

4.2 Experiments

4.2.1 Text clustering vs Markov chain clustering

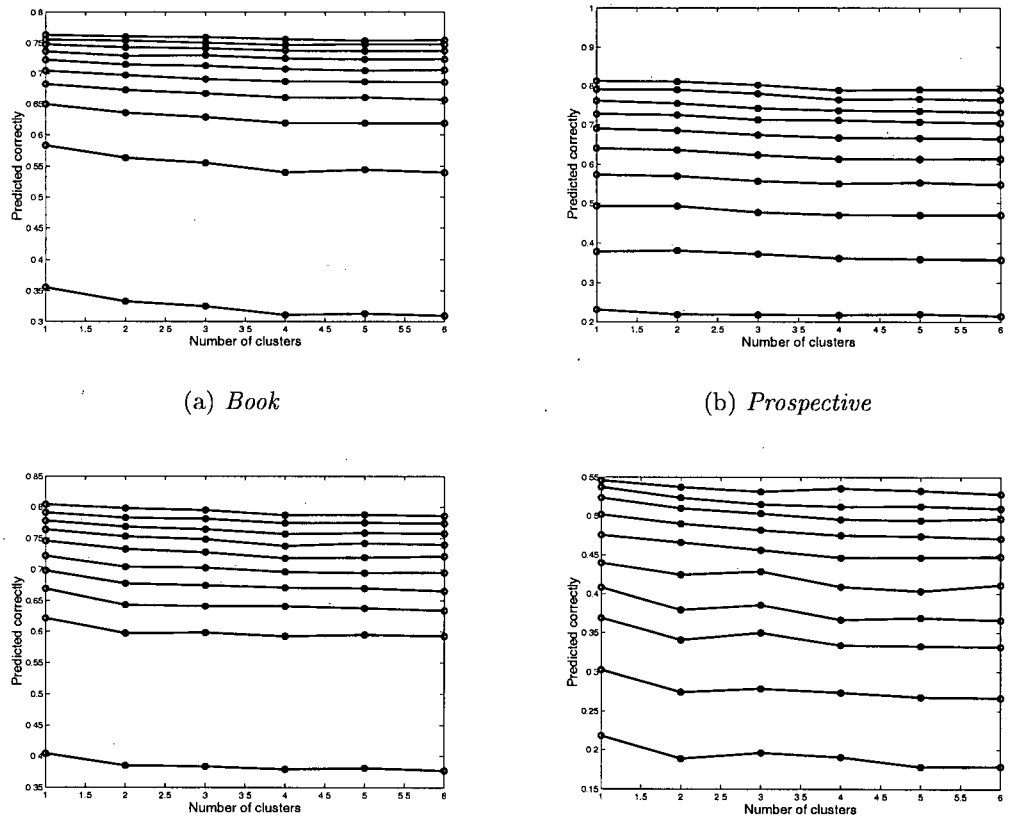
We compare text clustering to Markov chain clustering as the number of clusters varies from 1 to 6. We allow making multiple predictions, varying the size of prediction set from 1 to 10. We set $\alpha = 1$ and $\beta = 2$ for text clustering, and $\alpha = 1$, $\beta = 2$, and $\gamma = 2$ for Markov chain clustering.

Since EM clustering algorithms converge to a local, not global, maximum of likelihood function, different runs of Expectation-Maximization clustering result in different sets of clusters, some of which are inferior with respect to predictability.

We report maximal predictability over 10 runs of experiment. The rationale is that if we are able to find predictive clusters, we could detect them using a validation dataset, and then apply to the test dataset.

Figure 4.1 shows the predictability $p(Hit)$ for clustering sessions as texts.

From the bottom to the top, the 10 plots indicate $p(Hit)$ for the size of prediction set from 1 to 10 respectively.



(a) *Book* (b) *Prospective*

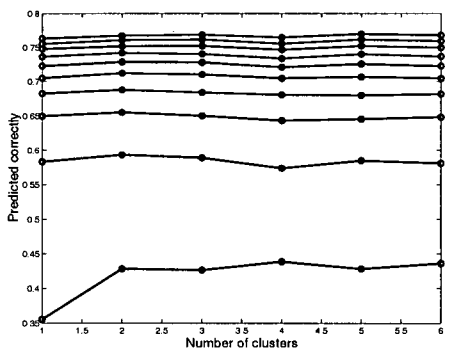
(c) *Ugrad* (d) *Grad*

Figure 4.1: Text clustering predictability for 1 to 10 predictions. From the bottom to the top, the 10 plots indicate $p(Hit)$ for the size of prediction set from 1 to 10 respectively.

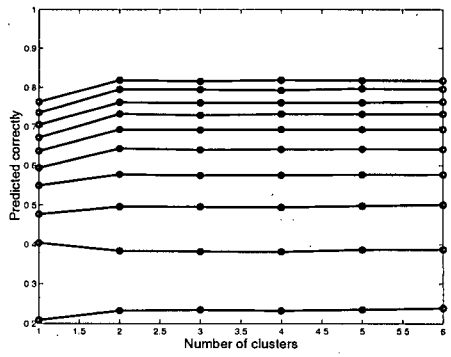
All datasets show decline in predictability as the number of clusters increases. This trend is common for the whole range of size of prediction set.

This indicates that clustering web sessions as texts is generally an inferior option for prediction. So, our assumption that the content of sessions will retain enough information needed for prediction does not hold.

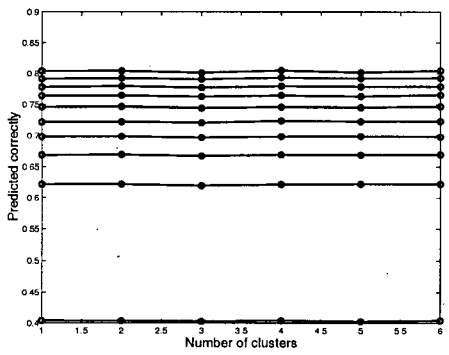
Figure 4.2 shows the respective $p(Hit)$ for clustering sessions as Markov chains. From the bottom to the top, the 10 plots indicate $p(Hit)$ for the size of prediction set from 1 to 10 respectively.



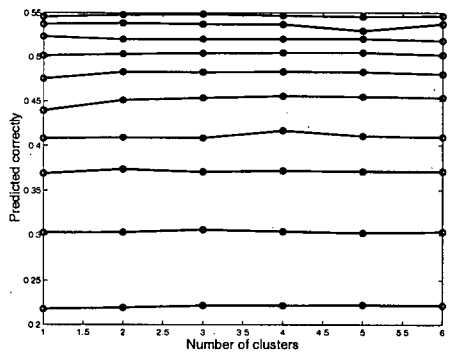
(a) *Book*



(b) *Prospective*



(c) *Ugrad*



(d) *Grad*

Figure 4.2: Markov chain clustering predictability for 1 to 10 predictions. From the bottom to the top, the 10 plots indicate $p(Hit)$ for the size of prediction set from 1 to 10 respectively

Markov chain clustering increases predictability for a limited range of prediction set sizes for the *Book* and *Prospective* datasets. For the other datasets, there is only a slight difference in predictability.

Markov chain clustering performed better than text clustering, yet it did not

Dataset	Basic model			No.Clusters				
	1st	All-2	All-3	2	3	4	5	6
Book	671	7732	17820	1342	2013	2684	3355	4026
Prospective	218	1316	3669	436	654	872	1090	1308
Ugrad	498	4193	12298	996	1494	1992	2490	2988
Grad	329	1872	4350	658	987	1316	1645	1974

Table 4.2: Number of states for different models

sufficiently improve predictability over the first order Markov model.

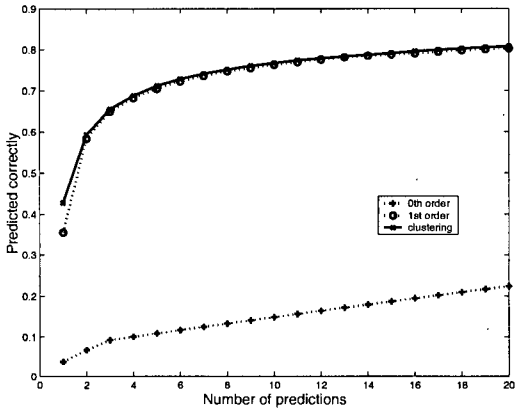
4.2.2 Markov chain clustering vs All-Kth Markov models

We compare the predictive power $p(Hit)$ of Markov chain clustering with two clusters to four baseline models as the size of prediction set varies from 1 to 20. Figures 4.3 - 4.6, plots (a) - (c) show the comparison of Markov chain clustering to the 0th and 1st, All-2nd, and All-3rd order Markov models with no clustering.

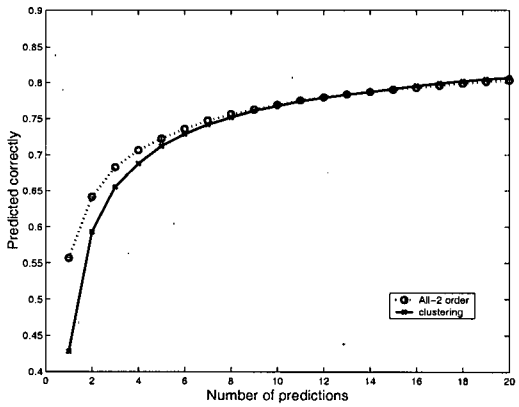
For the *Book* dataset the zeroth order model is completely inferior to the other models. For *Grad* and *Prospective* the performance ot the zeroth order model is comparable to that of the other models for lower sizes of prediction set, and is inferior otherwise. For the *Ugrad* dataset the zeroth order model is superior for a single prediction, and inferior for prediction sets larger than 2.

For all datasets the clustered model gets close to that of the higher order models starting with a certain size of prediction set. This is because when a prediction set is large enough, it covers nearly all possible user activities from the page, so higher order models cannot improve prediction further. For *Ugrad*, higher order models are inferior to the first order model for prediction sets smaller than 12.

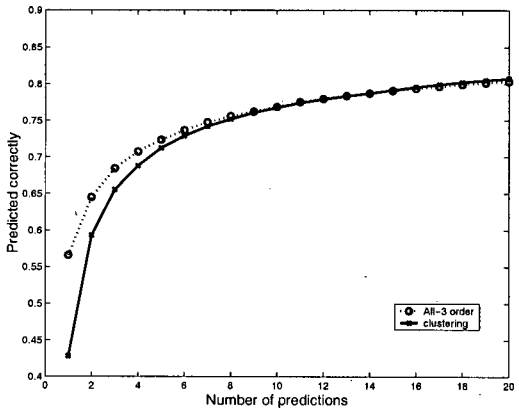
Table 4.2 presents the comparison of model complexity. For a clustered model, the number of model states is that of the first order model times the number of clusters. The complexity of clustered model is significantly lower than that of All-2 and All-3 Markov models.



(a)

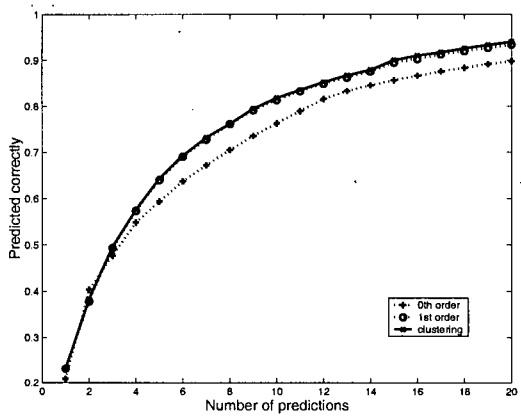


(b)

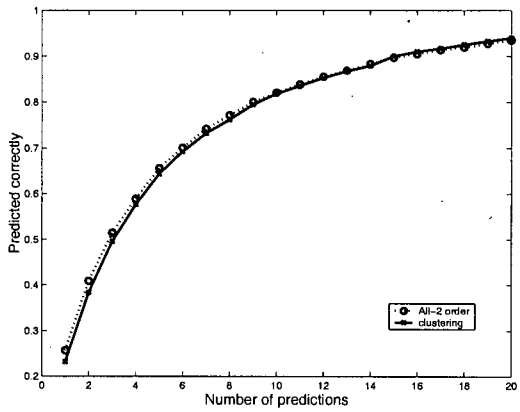


(c)

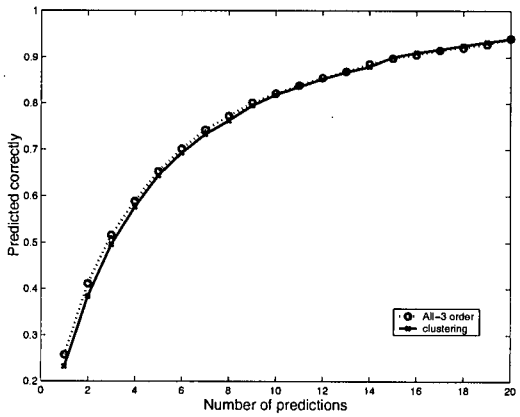
Figure 4.3: Markov chain clustering vs. plain prediction for the *Book* dataset



(a)

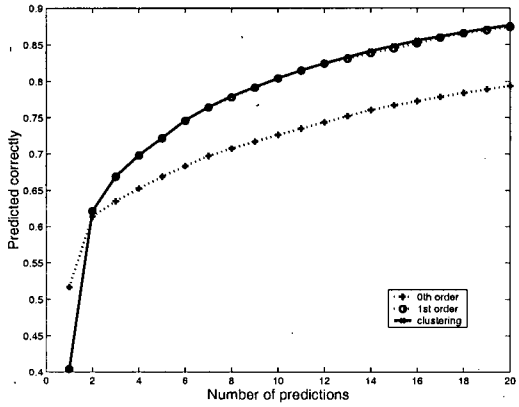


(b)

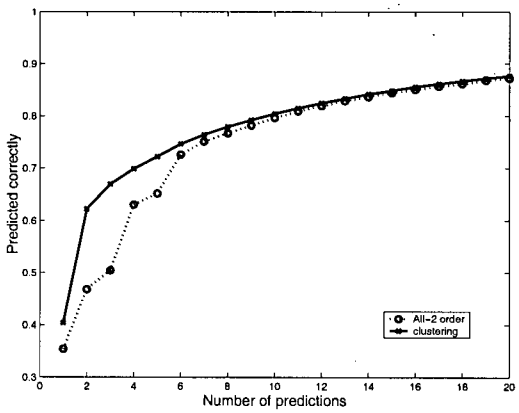


(c)

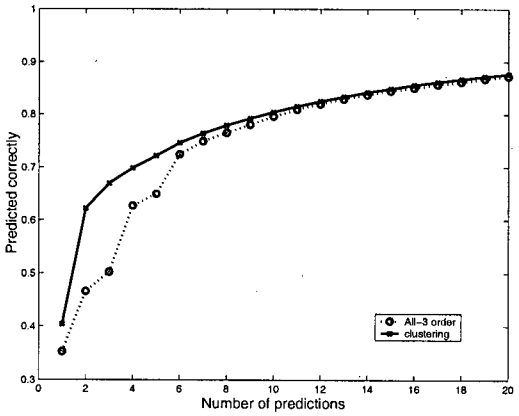
Figure 4.4: Markov chain clustering vs. plain prediction for the *Prospective* dataset



(a)

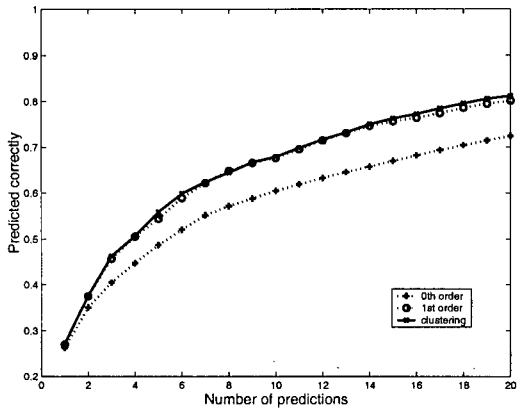


(b)

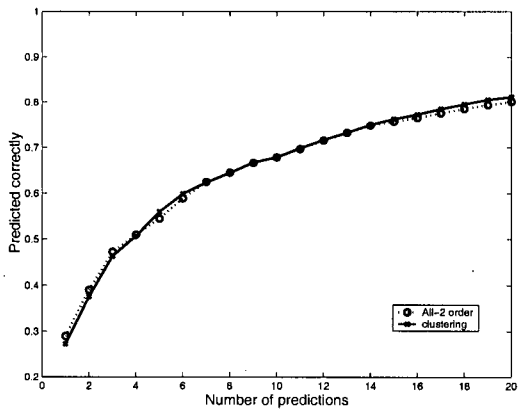


(c)

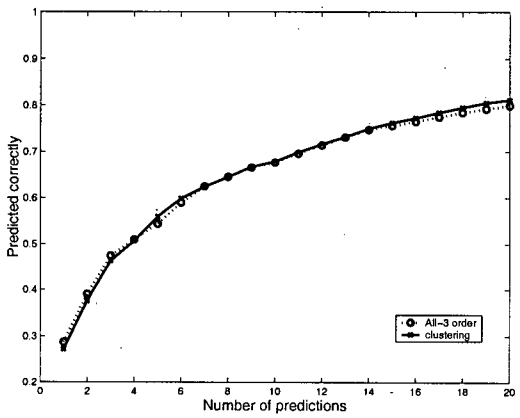
Figure 4.5: Markov chain clustering vs. plain prediction for the *Ugrad* dataset



(a)



(b)



(c)

Figure 4.6: Markov chain clustering vs. plain prediction for the *Grad* dataset

Chapter 5

Conclusions and Future Work

We considered the problem of predicting user's access to the web server. We studied performance of two predictive models with clustering. We clustered web sessions in two ways: as texts and as discrete Markov chains, and compared these models to non-clustered Markov models of orders 0, 1, All-2 and All-3.

We did the comparison across four subsets of Computer Science department web logs, namely, web pages for Graduate, Undergraduate, and Prospective students, and web pages for the Computational Intelligence textbook.

The experiments showed that the subsets had very different usage patterns. Hence there was no common ranking of baseline models and the clustered models over the subsets. So for a large heterogeneous web site a composite model may perform better than a uniform one.

Generally, Markov chain clustering performed better than text clustering, yet the former did not improve predictability over the first order Markov model.

Future work would include adding more attributes to web sessions for clustering, such as time spent by a user at a web page, distinguishing local and global users where appropriate, and building composite models for large heterogeneous web sites.

Bibliography

[1] Apache HTTP server Project, `http://httpd.apache.org/docs/logs.html`.

[2] A. Banerjee and J. Ghosh. Clickstream clustering using weighted longest common subsequences. In *Workshop on Web Mining at the First SIAM International Conference on Data Mining*, 2001.

[3] B. Berendt, B. Mobasher, M. Spiliopoulou, and J. Wiltshire. Measuring the accuracy of sessionizers for web usage analysis. In *Workshop on Web Mining at the First SIAM International Conference on Data Mining*, 2001.

[4] T. Berners-Lee, R. Fielding, and H. Frystyk. Hypertext Transfer Protocol – HTTP/1.0, Request for Comments RFC1945, 1996.

[5] A. Bestavros. Using speculation to reduce server load and service time on the WWW. Technical Report 1995-006, Computer Science Department, Boston University, 1995.

[6] I. V. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White. Visualization of navigation patterns on a web site using model-based clustering. In *Knowledge Discovery and Data Mining*, pages 280–284, 2000.

[7] R. Cooley, B. Mobasher, and J. Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1(1):5–32, 1999.

[8] N. de Freitas. Bayesian clustering of discrete Markov chains, 2002.

[9] M. Deshpande and G. Karypis. Selective Markov models for predicting web-page accesses. In *Workshop on Web Mining at the First SIAM International Conference on Data Mining*, 2001.

[10] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1, Request for Comments RFC2616, 1999.

[11] P. M. Hallam-Baker and B. Behlendorf. W3C Working Draft WD-Logfile-960323, `http://www.w3.org/tr/wd-logfile.html`.

[12] J. Han and M. Kamber. *Data Mining: Concepts and Techniques.* Morgan Kaufmann Publishers, 2000.

[13] T. Hofmann. Probabilistic latent semantic analysis. In *Proc. of Uncertainty in Artificial Intelligence, UAI'99*, Stockholm, 1999.

[14] A. Joshi and R. Krishnapuram. On mining web access logs. In *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 63–69, 2000.

[15] T. Palpanas. Web prefetching using partial match prediction. Technical Report CSRG-376, Graduate Department of Computer Science, University of Toronto, 1998.

[16] A. Papoulis. *Probability, Random Variables, and Stochastic Processes.* McGraw Hill, 1991.

[17] M. Perkowitz and O. Etzioni. Adaptive web sites: Automatically synthesizing web pages. In *AAAI/IAAI*, pages 727–732, 1998.

[18] P. Pirolli and J. E. Pitkow. Distribution of surfers' paths through the world wide web: Empirical characterizations. *World Wide Web*, 2(1-2):29–45, 1999.

[19] J. E. Pitkow and P. Pirolli. Mining longest repeating subsequences to predict world wide web surfing. In *USENIX Symposium on Internet Technologies and Systems*, 1999.

[20] D. Poole, A. Mackworth, and R. Goebel. *Computational Intelligence: A Logical Approach,* `http://www.cs.ubc.ca/spider/poole/ci.html`. Oxford University Press, 1998.

[21] G. Ridgeway. Finite discrete Markov process clustering. Technical Report MSR-TR-97-24, Microsoft Research, Redmond, WA, 1997.

[22] R.Sarukkai. Link prediction and path analysis using Markov chains. *Computer Networks*, 33(1–6):377–386, 2000.

[23] S. Schechter, M. Krishnan, and M.D.Smith. Using path profiles to predict HTTP requests. In *7th International World Wide Web Conference*, pages 457–467, 1998.

[24] R. Sen and M. H. Hansen. Predicting a web user's next access based on log data. *Journal of Computational and Graphical Statistics*, 2002.

[25] R. Srikant. Mining web logs to improve website organization. In *Proc. of 10th International World Wide Web Conference, WWW10*, Hong Kong, 2001.

[26] Z. Su, Q. Yang, Y. Lu, and H. Zhang. WhatNext: A prediction system for web request using n-gram sequence models. In *Web Information Systems Engineering*, pages 214–221, 2000.

[27] I. Zukerman, D. Albrecht, and A. Nicholson. Predicting users' requests on the WWW. In *UM99 – Proceedings of the Seventh International Conference on User Modeling*, 1999.