Model-Based Clustering and Visualization of Navigation Patterns on a Web Site

Igor Cadez[*], igor_cadez@sparta.com
David Heckerman[†], heckerma@microsoft.com
Christopher Meek[†], meek@microsoft.com
Padhraic Smyth[‡], smyth@ics.uci.edu
Steven White[†], stevewh@microsoft.com

Abstract

We present a new methodology for exploring and analyzing navigation patterns on a web site. The patterns that can be analyzed consist of sequences of URL categories traversed by users. In our approach, we first partition site users into clusters such that users with similar navigation paths through the site are placed into the same cluster. Then, for each cluster, we display these paths for users within that cluster. The clustering approach we employ is model-based (as opposed to distance-based) and partitions users according to the order in which they request web pages. In particular, we cluster users by learning a mixture of first-order Markov models using the Expectation-Maximization algorithm. The runtime of our algorithm scales linearly with the number of clusters and with the size of the data; and our implementation easily handles hundreds of thousands of user sessions in memory. In the paper, we describe the details of our method and a visualization tool based on it called WebCANVAS. We illustrate the use of our approach on user-traffic data from msnbc.com.

[*]Sparta Inc., 23382 Mill Creek Drive, #100 Laguna Hills, CA 92653
[†]Microsoft Research, Redmond, WA 98052-6399
[‡]School of Information and Computer Science, University of California, Irvine, CA 92697-3435

Keywords: Model-based clustering, sequence clustering, data visualization, Internet, web

1 Introduction

Arguably one of the great challenges for computer science in the coming century will be the understanding of human behavior in the context of "digital environments" such as the web. Given our limited experience to date with modeling such environments, there are relatively few existing theories or first-principles to guide any analysis or modeling endeavors. On the other hand, one can readily obtain vast quantities of data from such environments. In this context, data-driven exploration of digital traces—such as web-server logs—is certainly an important starting point for furthering our understanding of "digital behavior."

In this paper, we describe a novel approach to visualization and exploratory analysis of dynamic behavior of individuals visiting a particular web site. As a test bed for our work we use web-server logs of individual browsing records for many thousands of individuals or *users* at the msnbc.com site. Our approach is straightforward. First, we partition users into clusters such that users with similar behavior on the site are placed into the same cluster. Then, for each cluster, we display the behaviors of the users within that cluster.

The focus of our paper is on the clustering and visualization aspects of such data, rather than on the various engineering issues involved in preprocessing server-log data (identification, "sessionization," etc.). At this point, it is sufficient to assume a fairly abstract characterization of the data—that is, (a) the server-log files have been converted into a set of *sequences*, one sequence for each user session, (b) each sequence is represented as an

User	Sequence					
1	frontpage	news	travel	travel		
2	news	news	news	news	news	
3	frontpage	news	frontpage	news	frontpage	
4	news	news				
5	frontpage	news	news	travel	travel	travel
6	news	weather	weather	weather	weather	
7	news	health	health	business	business	business
8	frontpage	sports	sports	sports	weather	
9	weather					

Figure 1: A sample of user sequences.

ordered list of discrete symbols, and (c) each symbol represents one of several possible categories of web pages requested by the user. These categories correspond to sets of Uniform Resource Locators (URLs) on the site. Figure 1 shows a sample of such sequences. The web-servers from msnbc.com for a twenty-four-hour period typically produces roughly one million such sequences.

There are a number of aspects of the data which make the problem non-trivial. First, the information is inherently dynamic. Models and displays based on static information (such as histograms of pages requested) will not fully capture the dynamic nature of the underlying web-surfing behavior. Thus, we investigate relatively simple Markov models to represent dynamic behavior. An important point in this context is that these dynamic models serve primarily as vehicles for data exploration and we do not assume that the models necessarily represent the *true* data-generating process.

Second, dynamic behavior in this general context is highly likely to be quite heterogeneous. A population of users of this size will tend to have vastly different web-surfing

patterns in terms of (e.g.) the duration of a session and the content visited during a session. To address this heterogeneity, we imagine that different users lie in different *clusters*, where each cluster has a different Markov model. Specifically, we model the data as having been generated in the following fashion: (1) A user arrives at the web site and is assigned to a particular cluster with some probability, and (2) the behavior of that user is then generated from a Markov model with parameters specific to that cluster. We pretend this model generates the web data we observe, and that we only see the user behaviors and not the actual cluster assignments. We then use a standard learning technique, the Expectation–Maximization (EM) algorithm, to learn the proportion of users assigned to each cluster as well as the parameters of each Markov model. In so doing, we assign each user to a cluster or fractionally to the set of clusters. This approach to clustering is sometimes called a *model-based* (or mixture model) approach, and lies in contrast with the commonly used distance-based approaches. The clustering model we use is a finite mixture of Markov models. By using a model-based approach to clustering, sequences of different lengths may be assigned to the same cluster (e.g., sequence 1 and 5 in Figure 1 may very likely be generated from a single model corresponding to one of the clusters). This approach provides a natural and consistent mechanism for handling the problem of modeling and clustering sequences of different lengths. Full details of the model and the associated clustering algorithm are discussed in Section 2.

The third non-trivial aspect of the data is its size. Thus, it is critical that any algorithmic technique scale in a reasonable fashion—for example, linear or near-linear in the number of

clusters K, the number of sequences N, and the average number of category requests per sequence L. This requirement rules out—for example—any direct application of standard hierarchical clustering techniques that scale as $O(N^2)$ in both time and space complexity. An example of such an approach would be agglomerative clustering using (e.g.) some form of pair-wise edit-distance between sequences. In contrast, our algorithm for learning clusters of Markov chains (the EM algorithm) has a runtime per iteration that is $O(KNL + KM^2)$, where M is the number of different page categories that can be requested by a user. This complexity typically reduces to $O(KNL)$ for web data where M is relatively small. In Section 2.4, we investigate the overall runtime of our approach, and demonstrate by experiment that the total runtime of the algorithm (over all iterations) scales linearly in both N and K.

The paper proceeds as follows. Section 2 provides a detailed account of the underlying mixture model and the associated EM clustering algorithm, including experimental results on out-of-sample predictions comparing the quality of the Markov models with more traditional histogram approaches. In this section we also analyze the scalability of the overall clustering approach, using experimental results to validate the (often assumed) near-linearity of EM-based algorithms. In Section 3, we illustrate how the Markov clustering approach can be leveraged to support an interactive exploratory data analysis tool called WebCANVAS that provides direct insight into the heterogeneous and dynamic nature of this type of web data. Section 4 discusses why mixtures of Markov models are a useful model for navigation of categorized Web pages, even though non-mixture first-order Markov models

can be a poor model for navigation of uncategorized ("raw") Web pages. Section 5 briefly summarizes related work, and Section 6 concludes the paper with a summary and possible extensions of this work.

The primary novel contributions of this paper lie in (1) the introduction and evaluation of mixtures of Markov models for clustering and modeling of web navigation data, and (2) the use of the resulting clusters as the basis for interactive exploration and visualization of massive web logs.

2 Clustering Methods

In this section, we provide details of our clustering approach. For reasons discussed in the introduction, we concentrate on the model-based approach.

2.1 Model-Based Clustering

In the model-based approach to clustering, we assume that our data is generated as follows:

1. A user arrives at the web site and is assigned to one of K clusters with some probability, and

2. Given that a user is in a cluster, his or her behavior is generated from some statistical model specific to that cluster.

We assume that the data from different users are generated independently given the model (the traditional i.i.d. assumption). Statisticians refer to such a model as a mixture model with K components. Initially, of course, we do not have the model; we have only the data.

Nonetheless, we can apply standard statistical techniques to our data to learn our model—namely, (1) the number of components, (2) the probability distribution used to assign users to the various clusters, and (3) the parameters of each model component. Once the model is learned, we can use it to assign each user to a cluster or fractionally to the set of clusters.

To describe the mixture model more formally, we need some notation. We denote a variable by a capitalized token (e.g., X, X_i), and the state or value of a corresponding variable by that same token in lower case (e.g., x, x_i). We denote a set of variables by a bold-face capitalized token (e.g., $\mathbf{X}, \mathbf{X}_i$). We use a corresponding bold-face lower-case token (e.g., $\mathbf{x}, \mathbf{x}_i$) to denote an assignment of state or value to each variable in a given set. We use $p(x|y)$ to denote the probability that $X = x$ given $Y = y$. We also use $p(x|y)$ to denote a probability distribution for X given Y. Whether $p(x|y)$ refers to a probability or a probability distribution will be clear from context.

Now, let $\mathbf{X}$ be a multivariate random variable taking on values corresponding to the behavior of individual users. Let C be a discrete-valued variable taking on values $c_1, \ldots, c_K$. The value of C corresponds to the unknown cluster assignment for a user. A *mixture model for* $\mathbf{X}$ *with* K *components* has the form:

$$p(\mathbf{x}|\theta) = \sum_{k=1}^{K} p(c_k|\theta)\ p_k(\mathbf{x}|c_k, \theta) \tag{1}$$

where $p(c_k|\theta)$ is the marginal probability of the k^{th} cluster ($\sum_k p(c_k|\theta) = 1$), $p_k(\mathbf{x}|c_k, \theta)$ is the statistical model describing the distribution for the variables for users in the k^{th} cluster, and θ denotes the *parameters* of the model.

For the work described in this paper, we consider the special case where $\mathbf{X} = (X_1, \ldots, X_L)$ is an arbitrarily long sequence of variables describing the user's path through the site. The variable X_i takes on some value x_i from among the M possible page categories that the user could have requested. Thus, for example, the sequence $(x_1, x_2, \ldots, x_L)$ indicates that the user first requests x_1, then x_2, and so on. Note that, in our analysis, x_L is always the "end" state, which indicates that no additional pages were requested.

In our main approach to modeling this data, we assume that each model component is a *first-order Markov model*:

$$p_k(\mathbf{x}|c_k, \theta) = p(x_1|\theta_k^I) \prod_{i=2}^{L} p(x_i|x_{i-1}, \theta_k^T).$$

where θ_k^I denotes the parameters of the probability distribution over the initial page-category request among users in cluster k, and θ_k^T denotes the parameters of the probability distributions over transitions from one category to the next by a user in cluster k. This model captures (to some degree) the order of the user's requests. Specifically, it captures the user's initial request, the dependency between two consecutive requests, and—by virtue of the inclusion of the end state—the last category requested. In our work, each variable X_i is finite, $p(x_1|\theta_k^I)$ is a multinomial distribution, and $p(x_i|x_{i-1}, \theta_k^T)$ is a set of multinomial distributions.

There are many variations of this simple model. For example, using higher order Markov models for each component, we can capture dependencies beyond those in consecutive requests. As we shall discuss in Section 4, this extension appears to be unnecessary for modeling user traffic on msnbc.com.

Another variation of this model is the *zeroth-order Markov model* (also called a *unigram model*):

$$p_k(\mathbf{x}|c_k, \theta) = \prod_{i=1}^{L} p(x_i|\theta_k^M).$$

where θ_k^M denotes the parameters of the marginal distribution over category requests for a user in cluster k. Again, $p(x_i|\theta_k^M)$ is a multinomial distribution. This variation is useful when the site administrator does not care about the order in which the requests are made.

We note that, by using a model-based approach, it is straightforward to define alternative partitions of the user population. That is, for purposes of data exploration, we take the position that there is no "correct" model for clustering — each model captures different aspects of the data. A domain expert can consider different criteria of interest for partitioning users, translate these criteria to alternative models, and evaluate each model in terms of the usefulness of insights gained. In the case of clustering traffic on a web site, a site administrator may or may not feel that order of visits are important and use a first-order or zeroth-order Markov model, respectively.

As we shall describe in the following Section, we can learn a mixture model (K and the model parameters) given our data. Once the model is learned, we can use it to assign users to clusters as follows. Given the observed behavior $\mathbf{x}$ of a user, we can compute the probability distribution over the hidden variable C corresponding to the cluster assignment of the user by Bayes' rule:

$$p(c_k|\mathbf{x}, \theta) = \frac{p(c_k|\theta)\ p_k(\mathbf{x}|c_k, \theta)}{\sum_{j=1}^{K} p(c_j|\theta)\ p_j(\mathbf{x}|c_j, \theta)} \tag{2}$$

The probabilities $p(c_k|\mathbf{x}, \theta)$ are sometimes called *membership probabilities*. Once we have computed these probabilities, we can either assign the user to the cluster with highest probability—a *hard* assignment—or assign the user fractionally to the set of clusters according to this distribution—a *soft* assignment. Both types of assignment are commonly used in practice. As we shall see, in the current implementation of our visualization tool WebCANVAS, we use hard assignments.

2.2 Learning Mixture Models from Data

Let us begin by considering methods for learning the parameters of a mixture model with known number of components K, given training data $\mathbf{d}_{train} = \{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$. One possible criterion for doing so is to identify those parameter values for θ that maximize the likelihood of the training data:

$$\theta^{ML} = \text{argmax}_\theta \; p(\mathbf{d}_{train}|\theta) = \text{argmax}_\theta \; \prod_{i=1}^{N} p(\mathbf{x}^i|\theta)$$

where the second equality follows from our i.i.d. assumption. These parameters are often referred to as *maximum likelihood* or ML estimates. Alternatively, to encode prior knowledge about the domain and/or to smooth the ML estimates, one can introduce a *prior probability distribution* over the parameters, denoted $p(\theta)$. In this situation, a criterion for learning the parameters is to identify those parameters that maximize the posterior probability of θ given our training data:

$$\theta^{MAP} = \text{argmax}_\theta \; p(\theta|\mathbf{d}_{train}) = \text{argmax}_\theta \; p(\mathbf{d}_{train}|\theta) \; p(\theta)/p(\mathbf{d}_{train})$$

where the second identity follows by Bayes' rule. These parameters are often referred to as *maximum a posteriori* or MAP estimates. When used in conjunction with vague or non-informative priors, MAP estimates are smoothed (i.e., less extreme) versions of ML estimates (see, e.g., Good, 1965). In the work described in this paper, we learn MAP estimates for the parameters θ using diffuse Dirichlet priors with an effective sample size of 10^{-2}. (Neither the predictive nor visualization results are sensitive to variations in this effective sample size between 1 and 10^{-4}.) One exception is that, for the prior on the mixture weights, we used an improper Dirichlet prior with an effective sample size of zero. The priors are discussed in more detail in the Appendix.

We learn the parameters using the EM algorithm, an iterative algorithm that finds local maxima for the MAP (and ML) parameter estimates (e.g., Dempster, Laird, and Rubin, 1977). The algorithm chooses starting values for the parameters, and then iterates between an Expectation or E step and a Maximization or M step until the parameters values converge to stable values (as described in the next paragraph). In the E step of the algorithm, given a current value of the parameters θ, we fractionally assign a user with behavior $\mathbf{x}$ to cluster c_k using the membership probabilities given by Equation 2. In the M step of the algorithm, we pretend that these fractional assignments correspond to real data, and reassign θ to be the MAP estimate given this fictitious data. (See the Appendix for more details.) By iteratively applying the E step and M step, we monotonically improve the estimates of the model parameters θ, ensuring convergence (under fairly general conditions) to a local maximum of the posterior distribution for θ.

There are several reasonable choices for a convergence criterion. In our implementation, we say that the algorithm has converged when two consecutive iterations produce log likelihoods on the training data that differ by less than 0.01%. To initialize the EM algorithm, we use the *noisy-marginal* method of Thiesson, Meek, Chickering, and Heckerman (1999) (see the Appendix for details). Finally, when learning a particular model, we run twenty sets of initial parameters to convergence, and then use the value for θ that has the highest posterior probability. We have found that such "restarting" yields a small systematic improvement (roughly, 0.2%) in the log posteriors of the parameters.

In the remainder of this section, let us consider how to identify a good value for K. If these clusters are to be used for visualization, as they are in our application, a sensible method in principle for choosing K would be have a site administrator look at models having $K = 1$, $K = 2$, and so on, and choose directly. Because this approach is usually too time consuming, in practice, we choose the number of clusters by finding the model that accurately predicts N_t new ("test") cases $\mathbf{d}_{test} = \{\mathbf{x}^{N+1}, \ldots, \mathbf{x}^{N+N_t}\}$. That is, we choose a model with K clusters that minimizes the out-of-sample predictive log score:

$$\text{Score}(K, \mathbf{d}_{test}) = -\frac{\sum_{j=1}^{N_t} \log_2 p(\mathbf{x}^j | \theta^K)}{\sum_{i=1}^{N_t} length(\mathbf{x}^i)} \tag{3}$$

where θ^K is the MAP estimate of the parameters obtained from the training data, and $length(\mathbf{x}^i)$ is the length of the sequence for user i. Note that log scores in general have interesting properties and have been used extensively (Bernardo, 1979). Also note that this particular log score, which uses a base-2 logarithm and a length-of-sequence normalization, corresponds to the average number of bits required by the model to encode a category

request made by the user.

2.3 Application to Msnbc.com

We applied the learning techniques we have just described to a large Web navigation data set. The data comes from Internet Information Server (IIS) logs for msnbc.com and news-related portions of msn.com for the entire day of September, 28, 1999 (Pacific Standard Time). Each sequence in the data set corresponds to page views of a user during that twenty-four hour period. Each event in the sequence corresponds to a user's request for a page. Requests are not recorded at the finest level of detail—that is, at the level of URL, but rather, they are recorded at the level of page *category*. The categories, developed for site analysis prior to this investigation, are representative of the structure of the site. The categories are `frontpage`, `news`, `tech`, `local`, `opinion`, `on-air`, `misc`, `weather`, `health`, `living`, `business`, `sports`, `summary`, `bbs` (bulletin board service), `travel`, `msn-news`, and `msn-sports`. The number of URLs per category ranges from 10 to 5000. Although the time of each request is known, we model only the order in which the pages are requested. Furthermore, any page requests served via a caching mechanism were not recorded in the server logs and, hence, not present in the data. The full data set consists of approximately one million sequences (users), with an average of 5.7 events per sequence. The data are available online at `kdd.ics.uci.edu/databases/msnbc/msnbc.html`.

For various model types and various cluster sizes K, we learned models using a training set of 100,023 sequences sampled at random from the original one million. (Increasing the sample size did not appreciably change the resulting cluster model or the predictive

log-score). We then evaluated the models using the out-of-sample predictive log score in Equation 3 on a different sample of 98,687 sequences drawn from the original data. EM was run in the manner specified in Section 2.2. All runs, including those described in the next section, were performed on a desktop PC with a Pentium III Xeon processor running at 500MHz with enough memory to avoid paging. In our largest runs with $K = 200$, only 11.5Mb of memory was used.

Figure 2 shows the out-of-sample predictive log scores for first- and zeroth-order Markov models for various values of the number of clusters K. We see that the predictive accuracy of both models increases rapidly as K increases initially. For the zeroth-order Markov models, the predictive accuracy continues to increase substantially, although less rapidly, as K increases further. For the first-order Markov models, the predictive accuracy reaches what appears to be a stable limit around $K = 60$. Also note that, for values of K of practical interest ($K < 400$), the best zeroth-order model is worse at predicting out-of-sample data than the worst ($K = 1$) first-order model.

In learning clusters for this data using the first-order Markov mixture model, we observe an interesting phenomenon that is likely to occur for other domains. In particular, we find that some of the individual model components encode two or more clusters. For example, consider two clusters: a cluster of users who initially request category a and then choose between categories b and c, and a cluster of users who initially request category d and then choose between categories e and f. These two clusters can be encoded in a single component of the mixture model, because the sequences for the separate clusters do not

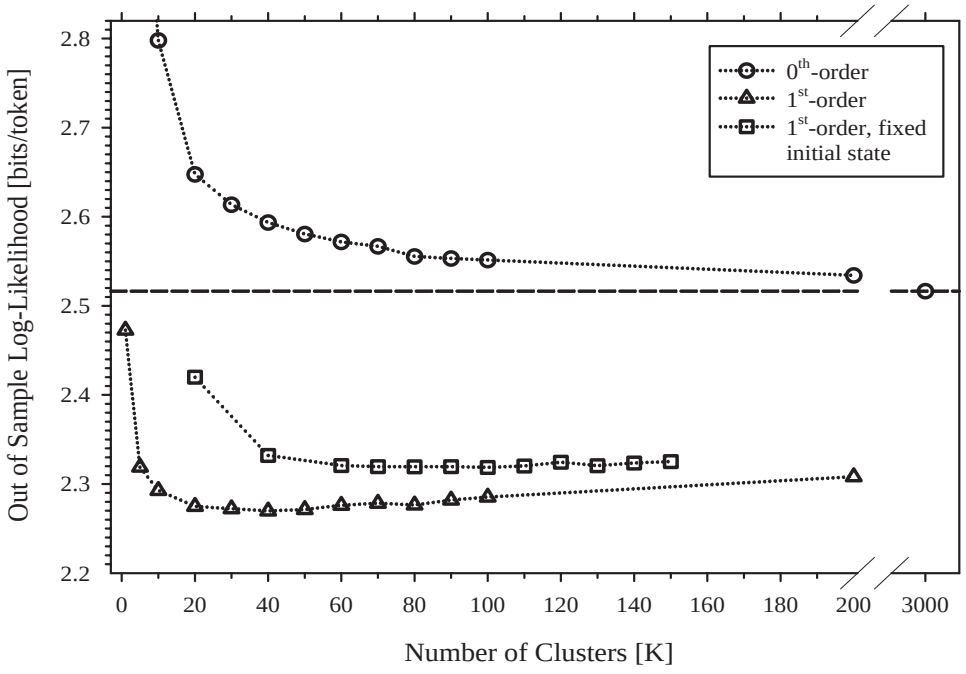

Figure 2: Number of bits (on average) needed to encode an out-of-sample event versus number of clusters K for a mixture of first-order Markov models, the same model constrained so that every user in the cluster has the same first page-category request, and a mixture of zeroth-order Markov models.

contain common elements.

The presence of multi-cluster components does not affect the out-of-sample predictive log score of a model. Nonetheless, when used in conjunction with our visualization tool, the existence of such components is problematic. Specifically, the behaviors of users from more than one cluster are presented in the same window, which can be confusing or distracting for visualization. Consequently, there is a need to produce models without multi-cluster components. One method to do so is to run the EM algorithm and then post-process the resulting model, separating any multi-cluster components found. A second method is to allow only one state (category) to have a non-zero probability of being the initial state in each of the first-order Markov models.

Using the second method can have the unfortunate consequence that a cluster of users that have different initial states but similar paths after the initial state are divided into separate clusters. Nonetheless, this potential problem was fairly insignificant for our msnbc.com data. In particular, Figure 2 shows the out-of-sample predictive log score for mixture models constrained to have the same first request. We see that these constrained models have a predictive power almost equal to that of the unconstrained models. Of course, when introducing this constraint, more components are needed to represent the data than in the unconstrained case. For this particular data, the constrained first-order Markov models reach their limit in predictive accuracy around $K = 100$, as compared to the unconstrained models, which reach their limit around $K = 60$. For our visualization in Section 3, we use the constrained model with $K = 100$ to assign users to clusters.

2.4 Scalability

As noted in the introduction, one of the inherent difficulties in the analysis of server-log data is in its size. In this section, we examine the scalability of the EM algorithm applied to our task of clustering sequence data.

The memory requirements of the algorithm are $O(NL + KM^2 + KM)$, which typically reduces to $O(NL)$—that is, the size of the data—for data sets where M is relatively small. In fact, our implementation can easily process hundreds of thousands of user sessions—all in memory—with main memory sizes that are typical for today's personal computers.

The runtime of the algorithm *per iteration* is linear in N and K. Nonetheless, it is difficult to mathematically characterize how the number of iterations required to reach convergence depends on N and K. When the number of sequences and particularly the number of clusters increase, the *shape* of the likelihood surface changes, and new local maxima or saddle points can appear. As a result, the number of iterations required for the algorithm to converge may increase with N or K.

To address this issue, we measured the *overall* runtime of the EM algorithm applied to our data set for various N and K. We varied the number of sequences from 10,000 to 200,000 and the number of clusters from 10 to 200. Each data point in the graphs represents the time required for our EM algorithm to converge. Figure 3 shows the overall runtime as a function of K and N. The dotted lines represent linear fits through the corresponding data points. These results demonstrate that, at least for the msnbc.com data set, the runtime of the EM algorithm scales linearly with N and K.

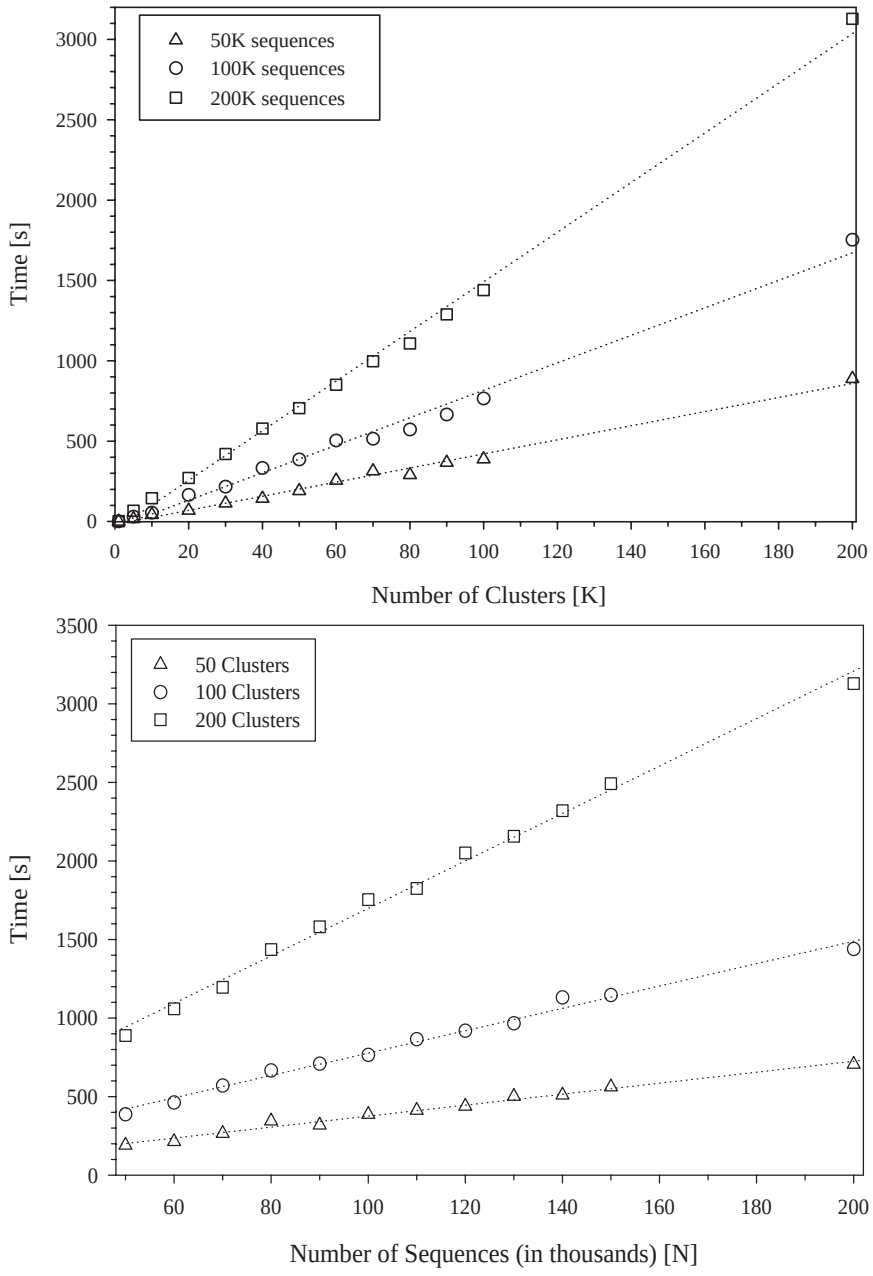

Figure 3: Running time of the first-order Markov Chain clustering algorithm. Dotted lines (linear fit) are included for reference.

3 Data Visualization

As discussed earlier, our approach to exploratory data analysis is to first cluster users and then visualize the behavior of users in each cluster. We have implemented a software tool that allows a site administrator to visually explore large sets of navigation sequences using the results of the clustering. The tool is called WebCANVAS (Web Clustering ANalysis and VisuAlization of Sequences).

In this section, we illustrate the visualization component of WebCANVAS using the msnbc.com data described earlier. We show clusters generated using a mixture of first-order Markov models applied to the same 100,023-sample described in Section 2.3. We note that the $K = 100$ clusters obtained from this sample did not change appreciably for larger samples.

Figure 4 shows WebCANVAS's initial display of twenty four of the one hundred clusters. The clusters normally are ordered left-to-right and top-to-bottom in descending order by the number of users in each cluster. This ordering provides useful information. For this display, however, we have scrambled the order of the clusters so as not to reveal potentially sensitive information about the msnbc.com site.

Each window corresponds to a cluster. The windows are tiled and each can be easily resized and/or scrolled. Each row of squares in a cluster corresponds to a user sequence. Note that WebCANVAS uses hard clustering, assigning each user to a single cluster. Each square in a row encodes a page request in a particular category encoded by the color of the square. (A category legend is shown in the lower-right corner of the screen.) For example,

the second user in the second cluster has the request sequence `news`, `on-air`, `on-air`, `local`, `opinion`, `opinion`, `on-air`, `opinion`, `news`. Note that the use of color to encode URL category limits the utility of this tool to domains where the number of categories can be limited to fifty or so.

In our experience with WebCANVAS, a site administrator can identify useful and unexpected information after only a few moments of looking at displays such as the one in Figure 4. In this case, the site administrator (S.W.) discovered several unexpected facts:

1. there were large groups of people entering msnbc.com on `tech` (clusters 11 and 13) and `local` (cluster 22) pages;
2. there was a large group of people navigating from `on-air` to `local` (cluster 12);
3. there was little navigation between `tech` and `business` sections; and
4. there were a large number of hits to the `weather` pages (cluster 1).

Each of these discoveries suggested actions to be taken to improve the site. For example, the unusually large number of hits to `weather` prompted an investigation of that portion of the site. The investigation revealed that response time was unacceptably slow (the many hits were due to users multi-clicking on weather pages in rapid succession in order to elicit a response from the site).

It is important to note that the visualization displays an (organized) sample of the raw data, where the choice of the sample is guided by a statistical model. We call this approach *model-directed sampling*. Although this approach is quite straightforward, we have found

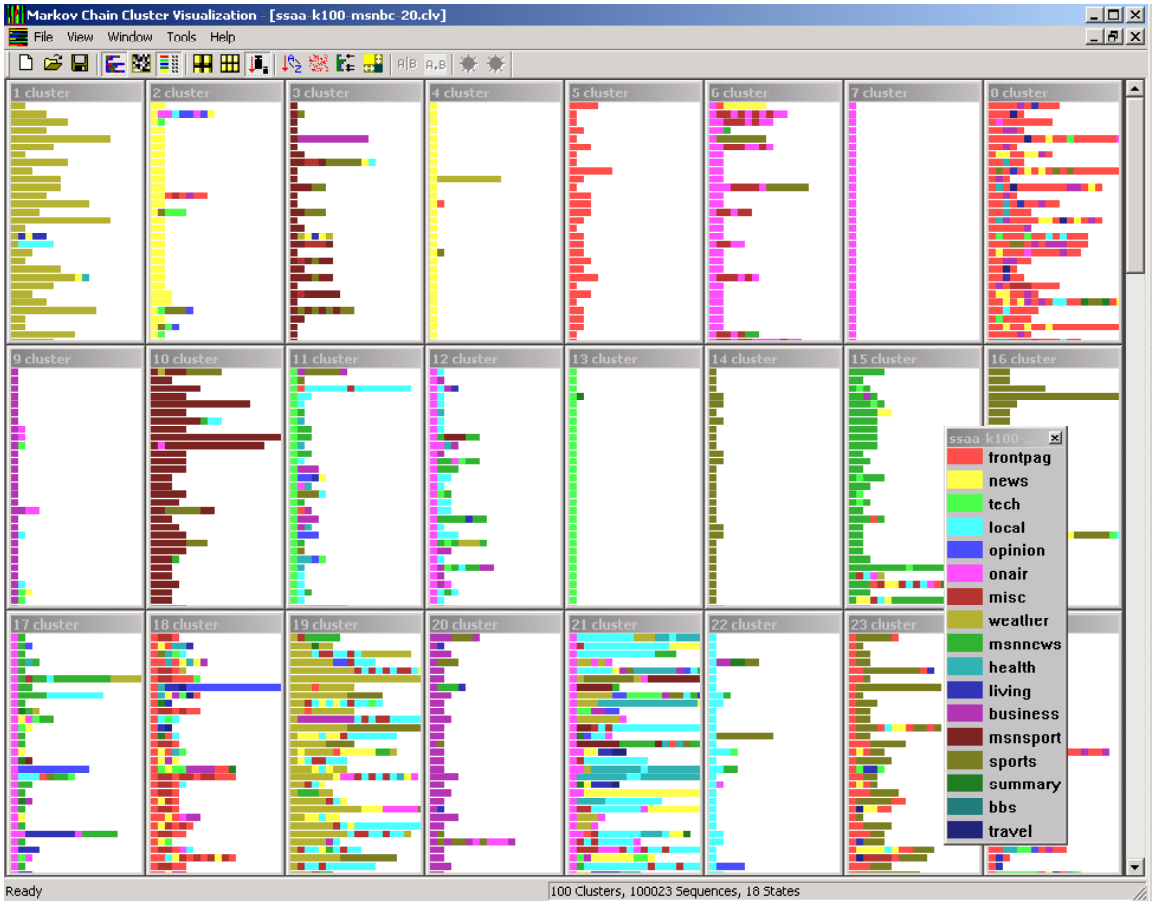

Figure 4: Initial display of msnbc.com data using WebCANVAS. Each window corresponds to a cluster. Each row in a window corresponds to the path of a single user through the site. Each path is color-coded by category. The category legend is at the lower right of the screen. (This figure appears in color in the online version of this paper).

it to be quite powerful in practice. In particular, we have found that people can gain an understanding of the behaviors in each cluster (both typical and deviations from typical) with a quick glance at each random sample.

In contrast, two other methods with which we experimented were not so transparent. In one approach, we showed the zeroth-order and first-order Markov models corresponding to a cluster as shown in Figure 5. In another approach, we used the "traffic-flow movie" produced by Microsoft Site Server v3.0. All of the authors tried the three approaches and unanimously found the model-directed sampling approach to be by far the most effective at communicating cluster content. Our informal experiments suggest that model-directed sampling is extremely effective for visualizing clusters, and may have more general application in data visualization.

One additional advantage of model-directed sampling over the second alternative of displaying the models themselves is that the former approach is not as sensitive to errors in modeling. That is, by displaying sampled raw data, behaviors in the data not consistent with the model used can still be seen and appreciated. In the next section, we shall provide evidence that our mixture of first-order Markov models is a reasonable model for web traffic (at least on msnbc.com), and thus this advantage may not play an important role in this particular application. Nonetheless, this advantage may be important in other domains.

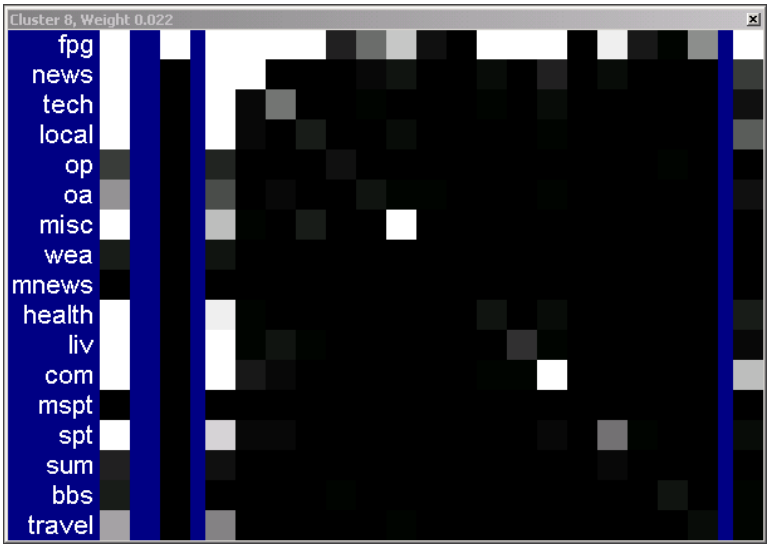

Figure 5: An alternative view of a cluster (cluster 8 in Figure 4). This view displays the zeroth-order and first-order Markov models for the cluster. Probabilities in the model are encoded by intensity (higher probabilities are brighter). The first column displays the zeroth-order distribution of category requests. That is, the first column shows, for each category, the probability that a user in the cluster will request a page in that category. The remaining columns show the first-order Markov model. The second column displays the probability distribution over category requests for the first event. The middle block displays the transition probabilities $p(j|i)$—the probability that a user will request category j given the previous request was category i. The last column shows, for each category, the probability that a user in the cluster will end his/her session given a visit to that category. In this particular display, users in the cluster start on `frontpage` and then may proceed to almost any other category. Once in a category, users may browse for a while, but return to `frontpage` before moving to another category. Users tend to leave the site from `frontpage` and `business`.

4 On the Suitability of Mixtures of First-Order Markov Models

The first-order-Markov mixture model that we use in this paper is quite simple. Indeed, there should be concern that this model is too simple. In this section, however, we describe experiments that suggest that the use of this model is appropriate—at least when applied to the msnbc.com data.

Before we do, we should make a connection with previous work (e.g., Sen and Hansen (2001), Deshpande and Karypis (2001)) that shows the first-order Markov model to be an inadequate model for empirically-observed page-request sequences. This result should not be surprising. For example, if a user visits a particular page, there tends to be a greater chance of he or she returning to that same page at a later time. A first-order Markov model cannot capture this type of "long-term" memory.

We make this connection because it is important to note that these previous results should not be used as evidence that the Markov mixture model is inadequate. First, although the mixture model is first-order Markov within a cluster, the overall unconditional model for Web navigation patterns is not first-order Markov. Second, the msnbc.com data is different from typical "raw" page-request sequences. Namely, our use of URL categories results in a relatively small alphabet size as compared to working with uncategorized URLs. The combined effects of clustering and a small alphabet tend to—at least for the msnbc.com data set—produce low-entropy clusters in the sense that a few (two or three) categories often dominate the sequences within each cluster. This effect is apparent in Figure 4. Thus,

the tendency to return to a specific page that was visited earlier in a session can be well approximated by our simple mixture of first-order Markov models, because the page categories visited *by a user in a given cluster* are typically constrained to visit the "dominant" categories for that cluster.

In Section 4.1, we emphasize the difference between our cluster model and the first-order Markov model. In Section 4.2, we describe diagnostic checks of our model.

4.1 Mixtures of First-Order Markov Models are not First-Order Markov

As in Section 2, let $\mathbf{x} = (x_1, \ldots, x_L)$ be a sequence of length L where each x_l is one of M values from a finite alphabet of symbols (the M page categories). We model $\mathbf{x}$ as being generated by a mixture of first-order Markov models:

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(\mathbf{x}|c_k)p(c_k) \tag{4}$$

$$p(\mathbf{x}|c_k) = p(x_1|c_k) \prod_{i=2}^{L} p(x_i|x_{i-1}, c_k)$$

Here, for simplicity, we have dropped the explicit mention of parameters.

It is informative to look at the predictive distribution for the next symbol x_{l+1} under this mixture model—that is, $p(x_{l+1}|x_{[l,1]})$, where $1 \leq l \leq L$, and $x_{[l,1]} = (x_1, \ldots, x_l)$. By definition,

$$\begin{aligned} p(x_{l+1}|x_{[l,1]}) &= \sum_{k=1}^{K} p(x_{l+1}, c_k|x_{[l,1]}) \\ &= \sum_{k=1}^{K} p(x_{l+1}|x_{[l,1]}, c_k)p(c_k|x_{[l,1]}) \\ &= \sum_{k=1}^{K} p(x_{l+1}|x_l, c_k)p(c_k|x_{[l,1]}) \end{aligned} \tag{5}$$

where the last line follows from the fact that, given component value c_k, x_{l+1} only depends on x_l. Thus, from Equation 5 above, the first-order-Markov mixture model defines the probability of the next symbol as a weighted convex combination of the transition probabilities $p(x_{l+1}|x_l, c_k)$ from each of the individual first-order component models. The weights are determined by the partial membership probabilities $p(c_k|x_{[l,1]})$ of the prefix ("history") subsequence $x_{[l,1]}$.

In contrast, the predictive distribution for a standard first-order Markov model (which can be viewed as a special case of the mixture model with $K = 1$) is simply

$$p(x_{l+1}|x_{[l,1]}) = p(x_{l+1}|x_l).$$

Comparing the mixture predictive model of Equation 5 with this standard first-order Markov model, we see that the two predictive distributions are not equivalent in the general case. In particular, a mixture of first-order Markov models leads to a predictive distribution that is itself not first-order Markov. More specifically, the transition probabilities in the mixture model are a function of the membership weights $p(c_k|x_{[l,1]})$. These weights are in turn a function of the history of the sequence (via Bayes rule), and typically depend strongly on the pattern of behavior before x_l.

As a specific example, consider the two sequences `AAAAAC` and `BBBBBC`. Consider the problem of predicting the next symbol. In a first-order Markov model, the predictive distribution on the next symbol will be the same for both sequences because it only depends on the current symbol `C`. In contrast, consider using a mixture of two Markov models, where cluster 1 produces sequences with long runs of `A`'s, no `B`'s, and an occasional `C`, and cluster

2 produces sequences with long runs of `B`'s, no `A`'s, and an occasional `C`. Under this mixture model, the prediction for what follows symbol `C` will be quite different for each of the sequences. Sequence `AAAAAC` is likely to have a probability near 1 of belonging to cluster 1, so the conditional (predictive) probability of `A` will be near 1. In contrast, the conditional probability of `B` will be near 1 for sequence `BBBBBC`.

Thus, a mixture of first-order Markov models is a semantically richer model than a non-mixture first-order Markov model. The mixture can represent higher-order information for predictive purposes because the weights can encode (in a constrained fashion) information from symbols that precede the current symbol.

As an aside, we note that our model evaluation score has an additional interpretation as a cumulative sum of the "one-step ahead" predictive distributions. To understand this point, consider the logarithm of the most general form of the distribution over a sequence $\mathbf{x} = (x_1, \ldots, x_L)$ of length L, and consider further its decomposition by the chain rule:

$$\begin{aligned}
\log p(\mathbf{x}) &= \log p(x_1, x_2, \ldots, x_L) \\
&= \log \Big[p(x_1)\ p(x_2|x_1)\ \cdots\ p(x_L|x_{[L-1,1]}) \Big] \\
&= \log p(x_1) + \log p(x_2|x_1) + \cdots + \log p(x_L|x_{[L-1,1]})
\end{aligned}$$

When this equation is applied to the whole data set $\mathbf{d}_{test}$ we obtain:

$$\log p(\mathbf{d}_{test}) = \sum_{i=1}^{N} \left[\log p(x_1^i) + \sum_{l=2}^{L_i} \log p(x_l^i | x_{[l-1,1]}^i) \right]$$

Therefore, maximization of the log-likelihood on test data is equivalent to optimizing the prediction of the next symbol.

Although the main focus of the work described in this paper is on clustering and visualization rather than "one-step ahead" predictive modeling, it is nonetheless worth pointing out that if the model *were* to be used for prediction, it is richer than a simple first-order Markov model because it possesses the capability of capturing some higher-order dependencies via the mixture mechanism. In fact, the experimental results presented in Figure 2 demonstrate that mixtures of first-order Markov models yield significant improvements in terms of out-of-sample prediction with respect to a simple first-order model for our domain. Anderson, Domingos, and Weld (2001) also found that mixtures of Markov models outperform the simpler first-order Markov models on a set of selected Web-page prediction tasks.

4.2 Model Diagnostics

As a diagnostic check on the adequacy of the first-order Markov assumption, one can empirically calculate the run lengths of page categories for several of the most likely clusters. If the data are being generated by a first-order Markov model, then the distribution of these run lengths will obey a geometric distribution and the empirical estimate of this distribution should appear geometric within sampling error. Conversely, if the empirical distribution of run lengths is decidedly non-geometric, then a non-Markov distribution is suggested.

Using the clustering methodology described in Section 3, we calculated the empirical distribution of run lengths for each category for the five most populous clusters (those with the five highest mixture weights). We used hard clustering to assign users to clusters, although we would have obtained almost identical results had we used soft clustering, because

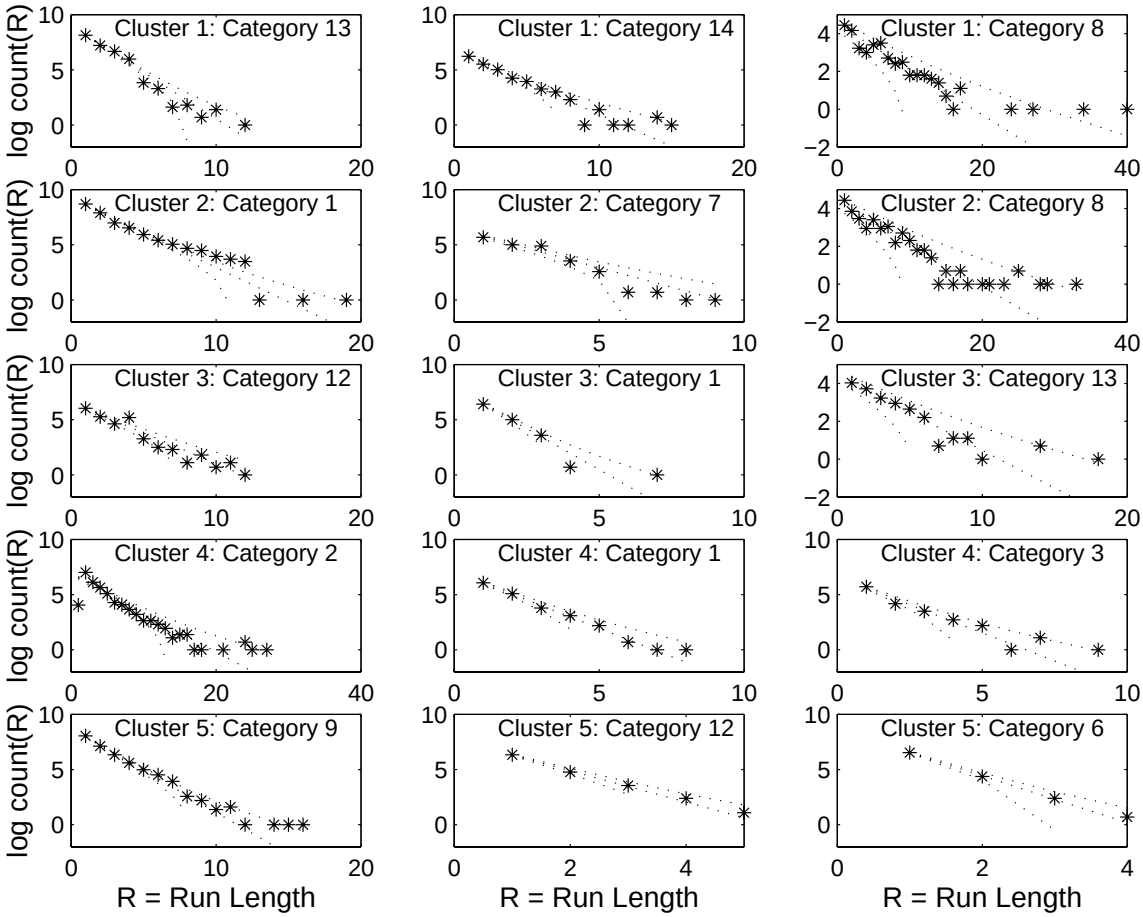

Figure 6: Empirically observed run lengths superposed on a geometric model for the three most requested categories in the five largest Markov clusters.

many of the cluster-membership probabilities were close to zero or one. To simplify our report of the results, we plot for each cluster the three most frequently visited categories that had at least one run length of four or greater. (Categories that have run lengths of three or fewer provide relatively uninformative diagnostic plots.)

The results are shown in Figure 6. The asterisks mark the empirically observed counts. The center dotted line on each plot is the expected count as a function of run length under

a geometric model using the empirically estimated self-transition probability of the Markov chain for the corresponding cluster. The upper and lower dotted lines represent the plus and minus three-sigma sampling deviations for each count under the model. With few exceptions (e.g., category 2, cluster 4), the geometric model is a reasonable approximation to the actual observed run lengths.

5 Related Work

Although there is a substantial amount of prior work on learning and modeling individual user navigation patterns from web data, much of this work is non-probabilistic in nature and focuses on finding rules that describe common navigation patterns (rather than clustering)—for example, Yan, Jacobsen, Garcia-Molina, and Dayal (1996), Chen, Park, and Yu (1998), Zaine, Xin, and Han (1998), Spilopoulou, Pohle, and Faulstich, (2000), Cooley, Tan, and Srivastava (2000), as well as numerous commercial systems.

There has also been some prior work that directly uses generative probabilistic models to characterize web site navigation patterns. Huberman et al. (1997) use a random walk approach to model the number of page-requests users issue at a particular web site. A variety of Markov models have been applied to the problem of prefetching pages for users conditioned on past pages visited—for example, Padmanabhan and Mogul (1996), Bestavros (1996) and Zukerman, Albrecht and Nicholson (1999). Pirolli and Pitkow (2000) investigated k-th order Markov models, with k ranging from one to nine, for modeling which link a typical user will follow from a given page. Borges and Levene (2000) also describe the use of

kth-order Markov models (in the form of probabilistic hypertext grammars) for web navigation modeling. Deshpande and Karypis (in press) reported improved predictive performance with high-order Markov models by selectively "pruning" certain state-dependencies in the model. These approaches all share with this paper the underlying use of a Markov representation for modeling of users' dynamic behavior, but focus on a single model for aggregate population characteristics, rather than learning clusters of behavior for different groups of users.

Sarukkai (2000) uses first-order Markov models to model the sequence of pages (or categories of pages) requested by a user. A "personalized" Markov model is trained for each different individual and then used for various prediction-related tasks for that user in future sessions. Sen and Hansen (2003) evaluated variations of second-order Markov and mixture models for page-request prediction for prefetching, where the mixtures here are mixtures of individual pages rather than of sequences. Anderson, Domingos, and Weld (2001) evaluated variants of first-order Markov models, conditional independence models, and mixtures of Markov models for the problem of predicting short-cuts in web page navigation and found that mixtures of Markov models generally had the best predictive performance. Although these papers use multiple Markov models (in various forms) to model page-request sequences, their primary focus is on prediction rather than on clustering and visualization.

In fact, the problem of clustering users based on their web navigation patterns has received little attention. Fu, Sandhu, and Shih (2000) applied BIRCH (a distance-based clustering algorithm) to clustering user sessions, where each session is represented as a vector

of times spent by the user on each page—that is, a static representation of user behavior.

In a general (non-web) context, the use of model-based probabilistic clustering for multivariate vector data is well known and widely used. For general reviews see Titterington, Smith and Makov (1985), McLachlan and Basford (1988), Banfield and Raftery (1993), Cheeseman and Stutz (1995), and Fraley and Raftery (1998). In addition, there have been numerous applications of this approach in areas as diverse as consumer marketing (Wedel and Kamakura, 1998) and atmospheric science (Smyth, Ide, and Ghil, 1999).

Nonetheless, there is relatively little work on probabilistic model-based clustering of sequences. Rabiner, Lee, Juang, and Wilpon (1989) provide an early algorithm for clustering different speech utterances using mixtures of hidden Markov models. Poulsen (1990) introduced a particular form of Markov mixtures for modeling heterogeneous behavior in consumer purchasing data. Krogh (1994) mentions the possibility of using mixtures of hidden Markov models for clustering sequences. More general versions of sequence clustering using Markov mixtures were independently developed by both Smyth (1997, 1999) and Ridgeway and Altschul (1998), including a general EM framework for learning. Cadez and Smyth (1999) have shown that all of these algorithms can be viewed as special cases of a general Bayesian hierarchical model. To our knowledge, the work reported here is the first application of sequence-based probabilistic clustering to web navigation data.

In terms of visualization of navigation patterns, there are numerous commercial (and often unpublished) systems that allow one to visualize user navigation patterns at a particular web site. These systems do not appear to use probabilistic dynamic cluster models. In

a similar vein, the Footprints work of Wexelblat and Maes (1999) provide a variety of techniques and interface tools which allow web surfers to enhance their "information foraging" experience by visualizing the history of other users using visual metaphors of maps, paths, and signposts. Minar and Donath (1999) use planar graphs to visualize "crowds of users" at particular web pages. This type of visualization work can be viewed as complementary to the more quantitative modeling approach we pursue here.

6 Summary and Future Work

We have developed a simple approach for clustering and visualizing user behavior on a web site, and implemented our method in a visualization tool called WebCANVAS. In our approach, we first cluster user behaviors using a mixture of first-order Markov models. We then display the behavior of a random sample of users in each cluster along with the size of each cluster. We have applied this approach to the visualization of web traffic on the msnbc.com site, and have found the approach to yield easy-to-understand, surprising, and useful insights.

In using first-order Markov models for clustering, we have taken into account the order in which each user requests pages. In fact, experiments described in this paper suggest that first-order Markov model mixture components are appropriate for the msnbc.com data. Another feature of our use of model-based clustering is that learning time scales linearly with sample size. In contrast, agglomerative distance-based methods scale quadratically with sample size.

Finally, there are several extensions to our approach that could be investigated. One is to model the *duration* of each visit. This extension can be achieved by using any number of duration models (e.g., log-normal). Another set of extensions avoid the limitation of our method to small M, modeling page visits at the URL level. In one such extension, we can use Markov models to characterize both the transitions among categories and the transitions among pages within a given category. Alternatively, we can use a hidden-Markov mixture model to learn categories and category transitions simultaneously.

7 Acknowledgments

We thank Max Chickering for his comments on an draft of this paper. The data set for this paper was provided by msnbc.com. The work of Igor Cadez was supported by a Microsoft Graduate Fellowship. The work of Padhraic Smyth was supported in part by the National Science Foundation under Grants IRI-9703120 and IIS-0083489, by an IBM Faculty Partnership award, and by a grant from Microsoft Research.

References

[Anderson et al., 2001] Anderson, C., Domingos, P., and Weld, D. (2001). Adaptive Web navigation for wireless devices. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 879–884. Morgan Kaufmann, San Francisco, CA.

[Banfield and Raftery, 1993] Banfield, J. and Raftery, A. (1993). Model-based Gaussian and non-Gaussian clustering. *Biometrics*, 49:803–821.

[Bernardo, 1979] Bernardo, J. (1979). Expected information as expected utility. *Annals of Statistics*, 7:686–690.

[Bernardo and Smith, 1994] Bernardo, J. and Smith, A. (1994). *Bayesian Theory*. John Wiley and Sons, New York.

[Bestavros, 1996] Bestavros, A. (1996). Speculative data dissemination and service to reduce server load, network traffic, and service time in distributed information systems. In Su, S. Y. W., editor, *Proceedings of the Twelfth International Conference on Data Engineering*, pages 180–187. IEEE Computer Society.

[Borges and Levene, 2000] Borges, J. and Levene, M. (2000). Data mining of user navigation patterns. In Masand, B. and Spiliopoulou, M., editors, *Web Usage Analysis and User Profiling*, pages 92–111. Springer, Berlin.

[Cadez and Smyth, 1999] Cadez, I. and Smyth, P. (1999). Probabilistic clustering using hierarchical models. Technical Report 99–16, Information and Computer Science, University of California, Irvine.

[Cheeseman and Stutz, 1995] Cheeseman, P. and Stutz, J. (1995). Bayesian classification (AutoClass): Theory and results. In Fayyad, U., Piatesky-Shapiro, G., Smyth, P., and Uthurusamy, R., editors, *Advances in Knowledge Discovery and Data Mining*, pages 153–180. AAAI Press, Menlo Park, CA.

[Chen et al., 1998] Chen, M.-S., Park, J., and Yu, P. (1998). Efficient data mining for traversal patterns. *IEEE Transactions on Knowledge and Data Engineering*, 10:209–221.

[Cooley et al., 2000] Cooley, R., Tan, P.-N., and Srivastava, J. (2000). Websift: the Web site information filter system. In Masand, B. and Spiliopoulou, M., editors, *Web Usage Analysis and User Profiling*, pages 163–182. Springer, Berlin.

[Dempster et al., 1977] Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, B 39:1–38.

[Deshpande and Karypis, 2003] Deshpande, M. and Karypis, G. (2003). Selective Markov models for predicting web-page accesses. *ACM Transactions on Internet Technology*. To appear.

[Fraley and Raftery, 1998] Fraley, C. and Raftery, A. (1998). How many clusters? Which clustering method? Answers via model-based cluster analysis. *Computer Journal*, 41:578–588.

[Fu et al., 2000] Fu, Y., Sandhu, K., and Shih, M.-Y. (2000). Clustering of Web users based on access patterns. In Masand, B. and Spiliopoulou, M., editors, *Web Usage Analysis and User Profiling*, pages 21–38. Springer, Berlin.

[Good, 1965] Good, I. (1965). *The Estimation of Probabilities*. MIT Press, Cambridge, MA.

[Huberman et al., 1997] Huberman, B., Pirolli, P., Pitkow, J., and Lukose, R. (1997). Strong regularities in World Wide Web surfing. *Science*, 280:95–97.

[Krogh, 1994] Krogh, A. (1994). Hidden Markov models in computational biology: Applications to protein modeling. *Journal of Molecular Biology*, 235:1501–1531.

[McLachlan and Basford, 1988] McLachlan, G. and Basford, K. (1988). *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker.

[Minar and Donath, 1999] Minar, N. and Donath, J. (1999). Visualizing crowds at a Web site. In *Conference on Human Factors in Computing Systems; CHI99*, pages 186–187.

[Padmanabhan and Mogul, 1996] Padmanabhan, V. and Mogul, J. (1996). Using predictive pre-fetching to improve world wide web latency. *ACM Computer Communication Review*, 26:22–36.

[Pirolli and J. Pitkow, 1999] Pirolli, P. and J. Pitkow, J. (1999). Distribution of surfer's paths through the world wide web. *World Wide Web*, 2:29–45.

[Poulsen, 1990] Poulsen, C. (1990). Mixed Markov and latent Markov modelling applied to brand choice behavior. *International Journal of Research in Marketing*, 7:5–19.

[Rabiner et al., 1989] Rabiner, L., Lee, C., Juang, B., and Wilpon, L. (1989). HMM clustering for connected word recognition. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 405–408. IEEE Computer Society Press, Los Alamitos, CA.

[Ridgeway and Altschuler,] Ridgeway, G. and Altschuler, S. Clustering finite discrete Markov chains. *Proceedings of the Section on Physical and Engineering Sciences*, pages 228–229.

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

[Sen and Hansen, 2003] Sen, R. and Hansen, M. (2003). Predicting a Web user's next access based on log data. *Journal of Computational Graphics and Statistics*, 12(1):143–155.

[Smyth, 1997] Smyth, P. (1997). Clustering sequences using hidden Markov models. In Mozer, M., Jordan, M., and Petsche, T., editors, *Advances in Neural Information Processing Systems 9*, pages 648–654. MIT Press.

[Smyth, 1999a] Smyth, P. (1999a). Multiple regimes in Northern hemisphere height fields via mixture model clustering. *Journal of the Atmospheric Sciences*, 56:3704–3723.

[Smyth, 1999b] Smyth, P. (1999b). Probabilistic model-based clustering of multivariate and sequential data. In *Proceedings of Seventh International Workshop on Artificial Intelligence and Statistics*, pages 299–304. Morgan Kaufmann, San Francsico, CA.

[Spiliopoulou et al., 2000] Spiliopoulou, M., Pohle, C., and Faulstich, L. (2000). Improving the effectiveness of a web site with Web usage mining. In Masand, B. and Spiliopoulou, M., editors, *Web Usage Analysis and User Profiling*, pages 142–162. Springer, Berlin.

[Thiesson et al., 1999] Thiesson, B., Meek, C., Chickering, D., and Heckerman, D. (1999). Computationally efficient methods for selecting among mixtures of graphical models, with discussion. In *Bayesian Statistics 6: Proceedings of the Sixth Valencia International Meeting*, pages 631–656. Clarendon Press, Oxford.

[Wedel and Kamakura, 1998] Wedel, M. and Kamakura, W. (1998). *Market Segmentation: Conceptual and Methodological Foundations*. Kluwer Academic Publishers.

[Wexelblat and Maes, 1999] Wexelblat, A. and Maes, P. (1999). Footprints: History-rich tools for information foraging. In *Proceedings of ACM CHI 99 Conference on Human Factors in Computing Systems*, pages 270–277.

[Yan et al., 1996] Yan, T., Jacobsen, M., Garcia-Molina, H., and Dayal, U. (1996). From user access patterns to dynamic hypertext linking. *Computer Networks*, 28(7–11):1007–1014.

[Zaiane et al., 1998] Zaiane, O., Xin, M., and Han, J. (1998). Discovering Web access patterns and trends by applying OLAP and data mining technology on Web logs. In *Proceedings of the Advances in Digital Libraries Conference*, pages 19–29.

[Zuckerman et al., 1999] Zuckerman, I., Albrecht, D., and Nicholson, A. (1999). Predicting user's requests on the WWW. In *Proceedings of the Seventh International Conference on User Modeling*, pages 275–284. Springer Wien.

A Appendix: The EM Algorithm for Mixtures of Markov Models

In this appendix, we describe the first-order-Markov mixture model and the associated learning algorithm in detail.

A.1 Notation and Model

As described in the main body of the paper, let $\mathbf{d}_{train} = \{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$ be a set of sequences, where each sequence $\mathbf{x}^i$ consists of L_i observed states $\mathbf{x}^i = (x_1^i, \ldots, x_{L_i}^i)$. Each state x_j^i takes values from a discrete alphabet $x_j^i \in [1, \ldots, M]$. In notation that is consistent with, but more detailed than that in the main body of the paper, we write $\theta = \{\pi, \theta^I, \theta^T\}$ where:

- π is a vector of K mixture weights:

$$\pi = \{\pi_1, \pi_2, \ldots, \pi_K\}, \;\; \pi_k = p(c_k|\theta), \;\; \sum_{k=1}^{K} \pi_k = 1.$$

- θ^I is a set of K initial state probability vectors:

$$\theta^I = \{\theta_1^I, \theta_2^I, \ldots, \theta_K^I\}$$

  where the per-component initial state probabilities $\theta_k^I, \;\; 1 \leq k \leq K$ are vectors of length M:

$$\theta_k^I = \{\theta_{k,1}^I, \theta_{k,2}^I, \ldots, \theta_{k,M}^I\}, \;\; \theta_{k,j}^I = p(x_1 = j|c_k, \theta), \;\; \sum_{j=1}^{M} \theta_{k,j}^I = 1.$$

- θ^T is a set of K transition matrices:

$$\theta^T = \{\theta_1^T, \theta_2^T, \ldots, \theta_K^T\}$$

where the per-component transition probability matrices $\theta_k^T,\ 1 \leq k \leq K$ are matrices of size $M-1 \times M$:

$$\theta_k^T = \left[\theta_{k,j,l}^T\right],\ \ \theta_{k,j,l}^T = p(x_t = l | x_{t-1} = j, c_k, \theta),\ \ \sum_{l=1}^{M} \theta_{k,j,l}^T = 1.$$

The probability of observing a particular sequence $\mathbf{x}^i$ under this K-component mixture model is therefore given by

$$\begin{aligned} p(\mathbf{x}^i|\theta) &= \sum_{k=1}^{K} p(c_k^i|\theta)\ p(\mathbf{x}^i|c_k,\theta) \\ &= \sum_{k=1}^{K} \pi_k\ p(\mathbf{x}^i|\theta_k^I,\theta_k^T) \\ &= \sum_{k=1}^{K} \pi_k\ p(x_1^i|\theta_k^I) \prod_{t=2}^{L_i} p(x_t^i|x_{t-1}^i,\theta_k^T) \\ &= \sum_{k=1}^{K} \pi_k \theta_{k,x_1^i}^I \prod_{t=2}^{L_i} \theta_{k,x_{t-1}^i,x_t^i}^T. \end{aligned} \tag{6}$$

The probability of observing a full data set $\mathbf{d}_{train} = \{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^N\}$ is known as the *likelihood* and is defined as

$$\begin{aligned} p(\mathbf{d}_{train}|\theta) &= \prod_{i=1}^{N} p(\mathbf{x}^i|\theta) \\ &= \prod_{i=1}^{N} \sum_{k=1}^{K} \pi_k \theta_{k,x_1^i}^I \prod_{t=2}^{L_i} \theta_{k,x_{t-1}^i,x_t^i}^T, \end{aligned} \tag{7}$$

where the product over the N sequences corresponds to an assumption that the individual sequences are mutually independent given θ (the i.i.d. assumption).

A.2 Prior Distributions

One approach to learning parameters from data is to find those parameter values that maximize the likelihood of the data. For mixture models, such maxima can not be found

in closed form. Consequently, iterative algorithms such as the EM algorithm to be outlined are used.

One difficulty associated with using this maximum-likelihood approach relates to zero probabilities. For example, suppose there are no transitions from A to B in the data. Then, our estimate of the transition probability from A to B in each mixture component will be zero. That is, according to our model, the transition is impossible. To address this difficulty, we can assign prior probabilities to θ, reflecting the belief that all transitions are possible, and use the maximum of the posterior distribution over θ as our estimate for the parameters.

The posterior distribution for θ given the data can be written (by Bayes' rule) as $p(\theta|\mathbf{d}_{train}) = p(\mathbf{d}_{train}|\theta)p(\theta)/p(\mathbf{d}_{train})$, where $p(\theta)$ is a prior distribution on θ. The maximum a posteriori (MAP) parameter vector is defined as the parameter vector θ that maximizes this posterior distribution, that is,

$$\theta^{\text{MAP}} = \arg\max_{\theta} p(\mathbf{d}_{train}|\theta)p(\theta)$$

where the term $p(\mathbf{d}_{train})$ is ignored since it is not a function of θ. Thus, the MAP parameters θ^{MAP} that correspond to the maximum of $p(\theta|\mathbf{d}_{train})$ can be found by maximizing the product of the likelihood and the prior, both viewed as functions of θ. In practice it is often convenient to work with the log of this expression, the log posterior function,

$$lP_{\mathbf{d}_{train}}(\theta) = \log p(\mathbf{d}_{train}|\theta) + \log p(\theta).$$

Again, for mixture models, closed-form solutions for θ^{MAP} do not exist; and iterative algo-

rithms (such as the EM procedure outlined in A.3) are used to search for maxima.

The parameters for the first-order-Markov mixture model (π, θ_k^I for every k, and each row of θ_k^T for every k) are discrete probability distributions with unknown values—that is, multinomial distributions. An often-used prior distribution for a multinomial distribution is the Dirichlet distribution. A Dirichlet distribution for the multinomial distribution with parameters $\phi = (\phi_1, \ldots, \phi_a)$ is given by

$$p(\phi_1, \ldots, \phi_a | \alpha_1, \ldots, \alpha_a) = \frac{\Gamma(\sum_{i=1}^a \alpha_i)}{\prod_{i=1}^a \Gamma(\alpha_i)} \prod_{j=1}^a \phi_j^{\alpha_j - 1},$$

subject to $\sum_{i=1}^a \phi_i = 1, \;\; 0 < \phi_i < 1, \;\; \alpha_i > 0$.

Given this Dirichlet prior for ϕ, suppose we observe data (a multinomial sample) such that there are n_i occurrences of state i for $i = 1, \ldots, a$. Then, the posterior distribution for ϕ is another Dirichlet distribution with hyperparameters $(\alpha_1 + n_1, \ldots, \alpha_a + n_a)$. In this regard, the Dirichlet distribution is said to be a *conjugate distribution* for multinomial sampling. Furthermore, the α's, which are often referred to as *hyperparameters* of the Dirichlet distribution, can thus be thought of as fictitious counts. Under this interpretation, the Dirichlet distribution is said to have an *equivalent sample size* of $\sum_{i=1}^a \alpha_i$.

The MAP values for any set of parameters will depend on the coordinate system used to express the parameters. The MAP values for a multinomial distribution expressed in the *natural parameter space* (see, e.g., Bernardo and Smith, 1994) is given by

$$\phi_i^{\text{MAP}} = \frac{n_i + \alpha_i}{\sum_{j=1}^a n_j + \alpha_j}, \quad i = 1, \ldots, a$$

A.3 Our EM-Algorithm Implementation

To describe the EM algorithm for finding a local maximum of our model parameters θ, it is convenient to define the *class-conditional probability distribution*—namely, the probability that sequence $\mathbf{x}^i$ was generated by cluster (or mixture component) k given parameters θ:

$$P_{i,k}(\theta) = p(c_k|\mathbf{x}^i, \theta) = \frac{\pi_k \; p(\mathbf{x}^i|c_k, \theta)}{\sum_{k'=1}^{K} \pi_{k'} p(\mathbf{x}^i|c_{k'}, \theta)}.$$

The set of class-conditional distributions for a data set can be represented by a matrix of probabilities

$$P(\theta) = [P_{i,k}(\theta)], \;\; 1 \leq i \leq N, \;\; 1 \leq k \leq K$$

where each row corresponds to the class-posterior for individual sequence $\mathbf{x}^i$.

A key quantity in the description of the EM is the expected value of the objective function over the class-posterior distribution using a fixed set of "current" parameters—the Q function. When this function is maximized with respect to the parameters, an update rule is derived that guarantees, under some weak conditions, that the objective function will increase and ultimately converge to a fixed point. The Q function for the log-posterior (MAP) function is defined as:

$$Q(\theta, \theta_{old}) = \langle lP_{\mathbf{d}_{train}}(\theta) \rangle_{P(\theta_{old})} = \sum_{i=1}^{N} \sum_{k=1}^{K} P_{i,k}(\theta_{old}) \log \Big[\pi_k p(\mathbf{x}^i|c_k, \theta) \Big] + \log p(\theta).$$

If we maximize the Q function with respect to each subset of parameters θ one can show that the following are the update rules for each set of parameters:

- Mixture Weights:

$$\pi_k = \frac{\sum_{i=1}^{N} P_{i,k}(\theta_{old}) + \alpha_k^{\pi}}{\sum_{k'=1}^{K}\left[\sum_{i=1}^{N} P_{i,k'}(\theta_{old}) + \alpha_{k'}^{\pi}\right]} \tag{8}$$

  where α_k^{π} is the hyperparameter associated with $\pi_k, k = 1, \ldots, K$. Note that this equation corresponds to computing MAP parameters as if the fractional assignment of data to the mixture components corresponded to real data. The same is true for the remaining update rules.

- Initial State Probabilities:

$$\theta_{k,j}^{I} = \frac{\sum_{i=1}^{N} P_{i,k}(\theta_{old})\delta(\mathbf{x}_1^i, j) + \alpha_{k,j}^{I}}{\sum_{j'=1}^{d}\left[\sum_{i=1}^{N} P_{i,k}(\theta_{old})\delta(\mathbf{x}_1^i, j') + \alpha_{k,j'}^{I}\right]} \tag{9}$$

  where $\alpha_{k,j}^{I}$ is the hyperparameter corresponding to $\theta_{k,j}^{I}$ and $\delta(\mathbf{x}_1^i, j)$ is an indicator function that is equal to 1 if the arguments are equal and 0 otherwise.

- Transition Probabilities:

$$\theta_{k,j,l}^{T} = \frac{\sum_{i=1}^{N} P_{i,k}(\theta_{old}) n_{j,l}(\mathbf{x}^i) + \alpha_{k,j,l}^{T}}{\sum_{l'=1}^{d}\left[\sum_{i=1}^{N} P_{i,k}(\theta_{old}) n_{j,l'}(\mathbf{x}^i) + \alpha_{k,j,l'}^{T}\right]} \tag{10}$$

  where $\alpha_{k,j,l}^{T}, l = 1, \ldots, M$ is the hyperparameter associated with $\theta_{k,j,l}^{T}$.

As mentioned in the main body of the paper, each multinomial distribution was assigned its own *uninformative* Dirichlet prior: $\alpha_k^{\pi} = 0$ for every k, $\alpha_{k,j}^{I} = 0.01/M$ for every k and j, and $\alpha_{k,j,l}^{I} = 0.01/M$ for every k, j, and l.

Also as mentioned in the paper, our implementation of the EM algorithm consists of one or more *runs*, where each run consists of an *initialization* phase and a *refinement* phase. In the initialization phase we select a set of random initial parameters (subject to

some constraints), whereas in the subsequent refinement phase, we iteratively apply the parameter update equations above to locally optimize the log-posterior function. Because the procedure is guaranteed only to find a local maximum of the log-posterior function, we perform several runs and report the solution with the highest value of the log posterior function. In our experiments, we perform twenty runs (each with different random initial parameter settings) with the following two phases:

- *Initialization.* We choose the parameters $\pi_1, \ldots, \pi_K$ to be equal. In addition, we use the noisy-marginal method of Thiesson et al. (1999) to initialize θ^I and θ^T. In this approach, we initialize the parameters for each component of the mixture model by estimating the parameters for a single-component cluster model, and then randomly perturbing the parameter values by a small amount to obtain K sets of parameters. In particular, we first determine the maximum-likelihood parameter configuration under the assumption that there is only one class. This step can be done in closed form. Next, for each multinomial distribution in this single-component model, we create a Dirichlet distribution such that (1) the maximum values of this Dirichlet distribution agrees with the corresponding maximum-likelihood estimates, and (2) the equivalent sample size of each Dirichlet is given by the user. In our experiments, we use an equivalent sample size of $2M$ for each Dirichlet distribution. We then sample the parameters for each mixture component from this Dirichlet distribution.

- *Refinement* consists of an iterative application of the update equations (8)–(10) until the relative change in the log-posterior function in consecutive iterations is less than

0.01%. In the experiments reported in this paper, the EM algorithm always converged in less than 100 iterations.