MIS Quarterly

SPECIAL ISSUE: BUSINESS INTELLIGENCE RESEARCH

A HIDDEN MARKOV MODEL FOR COLLABORATIVE FILTERING[1]

Nachiketa Sahoo
School of Management, Boston University, 595 Commonwealth Avenue, Boston, MA 02215 U.S.A.
and iLab, Heinz College, Carnegie Mellon University, Pittsburgh, PA 15213 U.S.A. {nachi@bu.edu}

Param Vir Singh
David A. Tepper School of Business and iLab, Heinz College, Carnegie Mellon University,
Pittsburgh, PA 15213 U.S.A. {psidhu@cmu.edu}

Tridas Mukhopadhyay
David A. Tepper School of Business and iLab, Heinz College, Carnegie Mellon University,
Pittsburgh, PA 15213 U.S.A. {tridas@cmu.edu}

In this paper, we present a method to make personalized recommendations when user preferences change over time. Most of the works in the recommender systems literature have been developed under the assumption that user preference has a static pattern. However, this is a strong assumption especially when the user is observed over a long period of time. With the help of a data set on employees' blog reading behavior, we show that users' product selection behaviors change over time. We propose a hidden Markov model to correctly interpret the users' product selection behaviors and make personalized recommendations. The user preference is modeled as a hidden Markov sequence. A variable number of product selections of different types by each user in each time period requires a novel observation model. We propose a negative binomial mixture of multinomial to model such observations. This allows us to identify stable global preferences of users and to track individual users through these preferences. We evaluate our model using three real-world data sets with different characteristics. They include data on employee blog reading behavior inside a firm, users' movie rating behavior at Netflix, and users' music listening behavior collected through last.fm. We compare the recommendation performance of the proposed model with that of a number of collaborative filtering algorithms and a recently proposed temporal link prediction algorithm. We find that the proposed HMM-based collaborative filter performs as well as the best among the alternative algorithms when the data is sparse or static. However, it outperforms the existing algorithms when the data is less sparse and the user preference is changing. We further examine the performances of the algorithms using simulated data with different characteristics and highlight the scenarios where it is beneficial to use a dynamic model to generate product recommendation.

Keywords: Recommender systems, collaborative filtering, changing preference, dynamic models, latent class model

[1]Hsinchun Chen, Roger Chiang, and Veda Storey were the accepting senior editors for this paper. Daniel Zeng served as the associate editor.

The appendices for this paper are located in the "Online Supplements" section of the *MIS Quarterly*'s website (http://www.misq.org).

Introduction

How do we generate personalized recommendations for users when their preferences are changing?

Motivation

Personalized recommender systems are used by online merchants to identify interesting products for their customers. This helps customers find the products they are likely to like from the thousands of items they would not have the resources to evaluate. It also enables merchants to focus their marketing efforts on advertising products to only those customers who might be interested in the products. Because of this, recommender systems have been used extensively at prominent large online stores such as Amazon.com. They also have generated tremendous interest in the research communities in information systems (Fleder and Hosanagar 2009; Sahoo et al. 2012), marketing (Ansari et al. 2000), and computer science (Resnick and Varian 1997).

The majority of recommender systems literature focuses on generating recommendations for users whose preferences are assumed to have static patterns (Adomavicius and Tuzhilin 2005). However, common experience suggests that user preferences can change over time. The changing preference is especially evident in cases where there is repeat consumption of experience goods of a certain class (e.g., music, news, movies, etc.). Consumers' preferences can change due to exposure to new kinds of products or due to the natural evolution of a person's taste (Koren 2010). This causes problems for a recommender system that has been trained to identify customers' preferences from their past ratings of products. Such a system might have successfully identified consumers' preferences in the past; however, recommendations made based on the estimated preferences may no longer be valid if the preferences change after the training period. In addition, there is a more serious problem encountered by the learning algorithms during the training phase. By fitting a static model to data generated by a dynamic process, one learns a mis-specified model. The system can produce the best average model that describes the user behavior. However, it would have little resemblance to the actual process generating the data and poor predictive power. Therefore, it behooves us to use a dynamic model when time-stamped user-ratings or user-purchase information are available. This is the motivation of the current paper.

Contribution

The key contribution of this paper is an approach to identify common patterns of change in user preferences and use them for more effective product recommendation. We examine the challenges in interpreting time-stamped ratings, and learning from them, when they are contributed by users whose unobserved preferences evolve over time. To overcome these challenges we present a hdden Markov model with a novel emission component. The model learns the global preference patterns that can be used to make personalized recommendations specific to particular time periods. The proposed algorithm is compared with the existing algorithms in the literature and the value of accounting for the users' changing preferences is demonstrated.

Literature Review

The current work relates to several streams of work in recommender systems, concept drift, and dynamic user behavior modeling. We review them selectively in this section to provide a context for this work.

Collaborative Filtering

One of the popular approaches to generate recommendations is collaborative filtering (Adomavicius and Tuzhilin 2005; Brusilovsky et al. 2007; Goldberg et al. 1992; Resnick et al. 1994a; Sarwar et al. 2001). Collaborative filters identify, from a list of items not seen by a user, those items the user is likely to like by analyzing the items other users of the system have rated. The inputs to the system are the records of data containing user ID, an item ID, and the rating the user has provided for the item. By providing ratings on items, users not only give the algorithm information about the quality of the items, but also about themselves (i.e., the types of items they like or dislike). The system outputs for each user a small set of items that the user has not seen before but is likely to like. This is in contrast to the content based filtering methods that recommend items with similar attributes to the items that a user has liked in the past (Lang 1995; Mooney and Roy 2000; Pazzani et al. 1996). The collaborative filtering algorithms have simpler data requirements. They do not need data on the properties of the items or demographic characteristics of the users. Unlike content-based approaches, collaborative filters are not limited to recommending only those items with attributes matching the items a user has liked in the past. Therefore, they have been popular in recommender systems.

The first group of collaborative filtering algorithms was primarily instance based (Resnick et al. 1994b). In the training step, these algorithms build a database of user ratings that is used to find similar users and/or items while generating recommendations. These algorithms became popular because they are simple, intuitive, and sufficient for many small data sets. However, they do not scale to large data sets without further approximations. Also, because they do not learn any user model from the available preferences, they are of limited use as data mining tools (Hofmann 2004).

A second group of collaborative filtering algorithms, known as model-based algorithms, surfaced later (Breese et al. 1998; Chien and George 1999; Getoor and Sahami 1999). They compile the available user preferences into compact statistical models from which the recommendations are generated. Notable model-based collaborative filtering approaches include singular value decomposition to identify latent structure in ratings (Billsus and Pazzani 1998), probabilistic clustering and Bayesian networks (Breese et al. 1998; Chien and George 1999), repeated clustering (Ungar and Foster 1998), dependency networks (Heckerman et al. 2001), latent class models (Hofmann and Puzicha 1999) and latent semantic models (Hofmann 2004) to cluster the ratings, and flexible mixture models to separately cluster users and items (Si and Jin 2003). Unlike the instance-based approach, model-based algorithms are slow to train, but once trained, they can generate recommendations quickly.

In recent years, the Netflix prize has provided new momentum to the research in collaborative filtering and recommender systems (Bell and Koren 2007; Koren 2009). The prize offered $1 million for developing an algorithm that predicts Netflix users' ratings on movies at least 10 percent more accurately than the existing system used by Netflix (Bennett and Lanning 2007). This has led to the development of many new algorithms. Some of the best performers among them are based on matrix factorization approaches (Koren et al. 2009; Paterek 2007). In these algorithms the *observed* user-item matrix is approximated by the product of a user factor matrix and an item factor matrix. The user factor matrix consists of columns of user weights (one column for each factor). Similarly, the item factor matrix consists of columns of item weights. The weights are the degrees of memberships of the users and the items into different latent factors.

Most of the collaborative filters are based on the assumption that a user's preference is a static pattern. The task of the filter is to learn this pattern so that it can predict the ratings the user will give to the items the user has not rated yet. The static assumption is a rather strong assumption, especially in certain classes of products that are used over a long time period. User preferences often evolve with the age of the user, changes in the user's work and social environments, or with the availability of new products. This leads to problems in estimating the model and predicting the items users are going to like.

The winning team of the Netflix prize, BellKor's Pragmatic Chaos, has shown that using *smooth functions* to model the *trends* of the users' average explicit rating on items leads to better estimation of the item ratings (Koren 2010). In another recent paper, user-specific Markov chains have been used to model the users' selection of items (Rendle et al. 2010). To alleviate the extreme data sparsity problem that one faces when estimating a transition matrix for each user, the authors use tensor factorization to isolate a few top factors that describe the dominant transition behavior. The paper highlights the need for recommending for a time period *after* the training period. However, while making the recommendation for the training period, it makes the implicit assumption that the user's preference is the same as the preference in the latest training period, which is inconsistent with the dynamic behavior assumption. Xiang et al. (2010) in a recent work have proposed a user–item–session graph to combine the long term preference of a user with the short term preference. The algorithm recommends based on a user/session-to-item proximity score on this graph. The time variable is used only to split a user's selection of items into different sessions. Thus, any ordering information in users' behavior is lost. In addition, since session definitions depend on users' selection of items, it cannot make any temporal recommendation for any session that has not already started.

There has been research in predicting link formation between one or more types of nodes. The techniques for predicting link between multiple types of nodes (e.g., users and products) can be used for product recommendation. A relatively new line of research in this area aims to use the time stamp of the links in the data set to make a more accurate prediction of links (Dunlavy et al. 2011). However, the temporal information is primarily used to discount the older data. There is evidence in other collaborative filtering research that this is not the best strategy (Koren 2010).

Despite recent research in incorporating various temporal elements in user ratings to make better recommendations, dynamic models of changing preferences remain a relatively less explored topic in collaborative filtering literature. One aspect of dynamic user behavior that is not currently modeled is the *patterns of changes* in user behavior from one time period to the next. This prevents one from predicting what the user preference will be at a time period after the training period. In this paper we attempt to fill this gap by taking a model-based approach that explicitly learns how user preferences change from one time period to the next. We also argue that instead of discounting the older data it is better to recognize that the older data might have been generated from a different preference of the user, and therefore can be used to learn about that user preference. This is beneficial for recommender systems because data from a user's past may not be useful for making recommendation for the user now, since her preference has changed, but it might be useful for making a recommendation for someone who currently has that preference.

Context-Aware Recommendation

Often the rating a user gives to an item depends on the context or need of the user at the time. This has led to a stream of research that models the user preference as dependent on context variables (Adomavicius and Tuzhilin 2010; Chen 2005; Van Setten et al. 2004). Some of the example applications include recommending an activity to a tourist depending on the location and the temperature of the day, or recommending movies to watch depending on the day of the week. Such systems can use two related strategies to produce recommendations.

One strategy is to slice the data so that each slice contains data specific to a given context. Then a separate system is trained for each context and the appropriate recommender system is used for the context for which the recommendations need to be generated. This often leads to data scarcity for each context-specific system. In a related second approach a distance measure is specified to determine the similarity between two contexts. This is used to determine how similar a context for a test scenario is to the contexts encountered during the training times. These distances are used to calculate a weighted combination of predicted scores of the individual recommender systems, which is then used to make a final context-aware recommendation.

Note that although these methods can produce two different recommendations under two different situations they estimate the user preference as static functions of environmental variables (i.e., a user's preference toward items is always determined in the same way from the context variables). In the current paper, we model the internal evolution of a user's preference over time in the absence of any knowledge of environmental factors.

Explicit Versus Implicit Ratings

A majority of the user feedback data used in collaborative filtering literature is in the form of user-ratings (i.e., after experiencing the item, the user tells us whether she liked the item or disliked it, and how much, by providing a rating on a scale of, for example, 1 to 5). However, there is a growing interest in developing algorithms for situations where the feedback is available only implicitly as a user's selection of an item (Hu et al. 2009; Pan and Scholz 2009). One of the primary motivations for developing such methods is that they pose virtually no cost to the user during the data collection. Since the data is simply the observation of users selecting certain items, such data is widely available (e.g., in click-streams present in the webserver access logs at online retailers; logs recording users' selection of programs to watch on their Internet Connected Television; at any brick-and-mortar store that keeps track of what its customers are buying through a membership program; at social bookmarking sites, such as delicious.com, that collect bookmarks shared by their members, etc.). There are several limitations of using such transactional data as implicit ratings (Hu et al. 2009). We observe a user selecting an item but we do not know if the user liked or disliked the item. Even in the collected bookmarks, where it may be assumed that the user bookmarked a webpage because the user liked it, when the entire data set consists of such bookmarks, we do not have any negative data points from which to learn. To simplify the scenario, often the selection is taken to be a positive rating (1) and lack of selection as a neutral rating (0) so that existing algorithms for explicit ratings could be applied. However, because of the outlined drawbacks of such a data set, the existing algorithms that are designed for explicit ratings do not work very well with implicit rating data. In addition, one has to be careful in how these algorithms are evaluated. Since these are not actual ratings, rating prediction errors such as mean square error and mean absolute errors are not appropriate. Instead the item retrieval performance metrics such as precision, and recall have been used to compare algorithms that use implicit ratings (Huang et al. 2007a).

Concept Drift

When observed data is generated from a distribution that changes over time, it is known as *concept drift* (Tsymbal 2004). Concept drift is observed in many phenomena of general interest (e.g., weather prediction rules differ from one season to the next). Market conditions and moods often have yearly or even monthly recurring patterns. The nature of spam e-mail has been shown to drift over time (Cunningham et al. 2003). Statisticians and machine learning researchers have long been interested in estimating models from data with concept drift that can be used for making reliable predictions in the next time period (Schlimmer and Granger 1986; Widmer and Kubat 1996). The strategies adopted can be summarized into two groups.

The first strategy is to discount the data that are not relevant for prediction. For example, old data can be weighted less or even excluded from the data set when estimating a predictive model. Other qualities of the data (e.g., noise, relevance, or redundancy) can also be used to weight the data (Cunningham et al. 2003).

The second strategy is to build an ensemble of models, each of which is fitted to a different subset of the data (Harries et al. 1998; Street and Kim 2001). This strategy has been applied in the topic detection and tracking initiative for identifying new news topics and tracking stories that occur in them. Such approaches maintain a finite number of models. The algorithms often rely on heuristics based on quality metrics to add a new model, update the existing models, or delete the outdated concepts if the nature of the data changes.

The key focus in most of the learning algorithms under concept drift is to keep the learned model current by weighting down the outdated data and models. However, as argued and demonstrated in a recent paper, for collaborative filtering applications, the loss of information from discarding old data often outweighs any benefit from the removal of irrelevant data (Koren 2010).

Dynamic Models

There is a rich stream of literature on statistical dynamic models. The two most closely related are hidden Markov models (HMM) and Markov switching models. An HMM is a model of a stochastic process that cannot be observed directly but can only be viewed through another set of stochastic processes that produce a set of observations (Rabiner 1989). One of the simplifying assumptions of the HMM is that the observed variable in a given time period is assumed to only depend on the value of the hidden variable in that time period. HMMs have been widely applied in speech recognition (Juang and Rabiner 1991), cryptanalysis (Karlof and Wagner 2003), part-of-speech tagging (Cutting et al. 1992), machine translation (Deng and Byrne 2008), gene finding (Lukashin and Borodovsky 1998), alignment of bio-sequences (Notredame 2002), software developer learning (Singh et al. 2006), and customer relationship management (Netzer et al. 2008). There have been numerous modifications to the original hidden Markov model. Some of the notable models include the variable duration hidden Markov model in which the number of steps the process can stay in a given state is modeled explicitly (Levinson 1986). Yet another variation of HMM developed for automated speech recognition, known as the *segment model*, considers sequences of varying length observed every time the process assumes a new state (Ostendorf et al. 1996). A detailed survey of the literature can be found in Murphy (2002).

The Markov switching models differ from the HMM in that it relaxes the assumption that the observed variable only depends on a hidden variable of the same time period to allow possible additional dependence on the observed variable of the previous time period. Modeling of such additional dependency makes the Markov switching model more suitable for modeling time series (Lu et al. 2010). This technique has been used to model many economic phenomena including identification of macroeconomic business cycles (Hamilton 1989) and modeling changing interest rates (Dahlquist and Gray 2000).

Despite this rich body of literature in dynamic models, there has been little research on examining user ratings in such a framework so that changes to user preferences can be inferred and used for generating more relevant recommendations. This paper aims to fill this gap.

Problem Definition

In this section, the problem is described and motivated in the context of a corporate blog network. Later we evaluate the proposed method on a data set collected from this blog network and two additional real world data sets, namely the Netflix prize movie ratings and the last.fm music listening records.

Context, Data Set and the Task

The motivation for this research comes from an observation in a corporate blog network. The increasing adoption of Web 2.0 technologies and cultures within enterprises is encouraging employees in firms to be content producers. This has resulted in large knowledge bases created by employees in the form of corporate blogs and wikis. This is an asset for firms because it gives employees access to the expertise of other employees. However, it also creates an information overload. Because there could be thousands of articles written by employees, finding the relevant article for a particular employee is not a trivial task. In this research, we have worked on the corporate blog network of a large Fortune 500 IT services firm and have proposed a recommender system to alleviate the information overload.

We collected the log of users' visits to blog articles from the webserver that hosts the blogs. This access log provided us with implicit ratings of users on the blog articles. The data set was collected over 22 months (January 2007–October 2008). There were 71,500 articles posted during this period. The articles were read by 52,000 employees. There are 2.2 million time stamped visits by the blog readers to the blog articles.

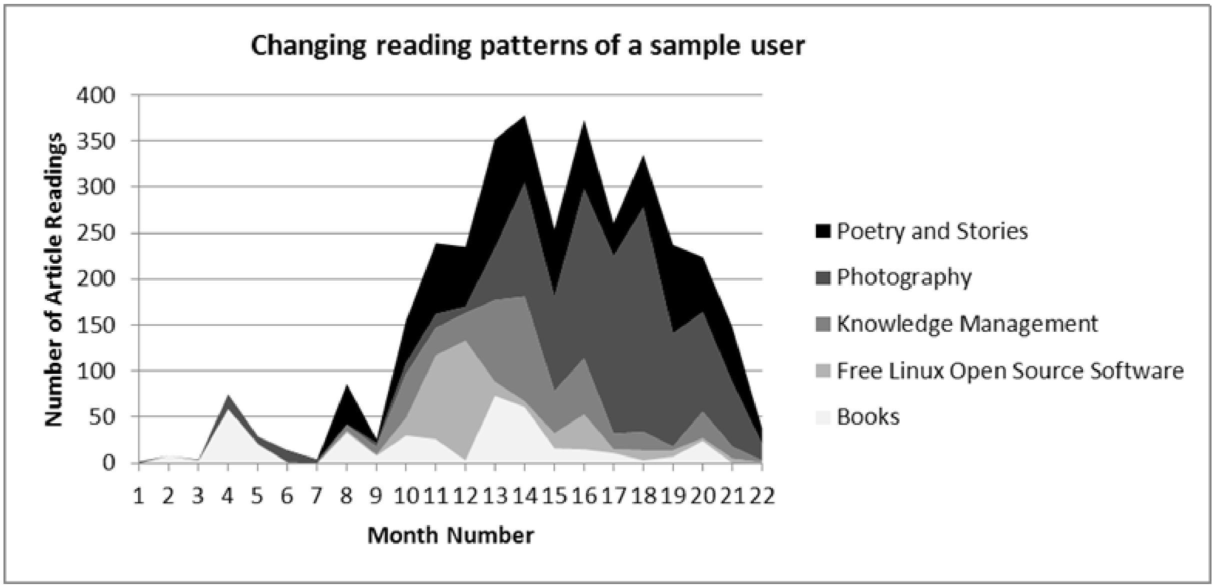


Figure 1. A Stacked Plot of the Number of Articles Read by a Sample User in Five Different Topics Over the Months

The articles have been classified into 25 predefined topics by their authors, including "Knowledge Management," "Software Testing," and "Photography."

An examination of the blog reading behavior shows that blog readers change the amount and type of posts they read over time. In Figure 1, we show the volume of articles read by a randomly selected user in 5 different topics over the 22 months. There seems to be a distinct change in the user's reading behavior over time. In addition to the increase in article reading around the center of the data collection period, the type of posts the user reads also changes. In months 4 and 5, the user was primarily reading blog articles about "Books." The user continues to read intermittently in this topic for most of the observation period. Later, around months 10 and 11, the user starts reading in the topic of "Linux and Open Source Software" and also in "Knowledge Management" and "Poetry and Stories." Although subsequently the user reduces reading in the topic of "Linux and Open Source Software," the reader continues to read articles in "Poetry and Stories" and slowly reduces reading "Knowledge Management" related posts. Around months 12 and 13 the user starts reading Photography related posts, which continues to dominate most of his/her reading activity in the subsequent months.

We define the *preference* of a user to be a latent property of the user, susceptible to influence from her environment, which leads her to select certain types and numbers of posts. The observation in Figure 1 suggests that the user's reading preference is not static but changing. Therefore, the assumption of static user preference by collaborative filters seems rather strong.

The changing preference also suggests that the task of practical importance is to make personalized recommendations for a *given time period*. This is a harder problem than the two static recommendation tasks often undertaken in the literature. In one class of tasks, the data is randomly divided into a training set and a test set. This potentially includes data from each time period in both sets. Thus, it provides evidence on a user's preferences in time periods from which the test data was collected. However, in practice we only have data from the past to use to predict the ratings in the future, when the user's preferences might be different. In a second class of tasks, the data is divided into those collected in *two* non-overlapping time periods. The data collected during the earlier period is used for training and the data collected from the later period is used for testing. Although this is more realistic than the previous scenario, the task to be solved in real life is harder (i.e., predicting whether a user is going to select an item in the next time period of certain finite length, not at any time after training). It means that during evaluation, the algorithm must successfully identify a smaller set of items that the user selects in a given test period.

In this paper our task is to recommend articles to users for *one time period* following the training period. This is closer to what a practitioner would use. We evaluate the algorithms for their performance in a time-specific recommendation.

Problems with the Static Model of User Preference

Let us now consider the user–user similarity based static collaborative filtering algorithm (Shardanand and Maes 1995). Each user's fixed preference is represented by the ratings she has provided on a set of items. Then the similarities between pairs of users are computed, so that items liked by similar users can be recommended to a target user. This approach breaks down when the preferences change over time. The rating data for each user is not generated from one fixed unknown preference, but from a series of unknown preferences. Therefore, it is not clear if one should find other similar users and recommend the items they have rated highly. The users are no longer identified with their changing preferences, and preferences ultimately determine whether a user likes an item.

A similar challenge exists for static model based collaborative filtering algorithms such as the aspect model (Hofmann and Puzicha 1999) (Figure 2). The aspect model is a probabilistic matrix factorization approach. In this model, the user preference is represented as a membership of the user in different latent classes to different degrees. For each user, this set of static class memberships uniquely defines her preference. In addition, each item belongs to different latent classes to different degrees. This set of memberships characterizes the item. The static model based algorithms are able to estimate the class memberships of users and items because of the assumption that all selections of the items by users are generated by the same set of class memberships of the user. However, this is too strong an assumption when the preferences of the users are changing.

Research Questions

The existing issues lead us to a set of research questions that need to be addressed to build personalized recommender systems for changing user behavior.

1. How can the old user ratings, generated by a prior temporary user preference, be used to learn user preference models that can be used to recommend items to users?

2. How can we learn a change in a user's preference from her unique ratings on items?

3. How do we model the behavior of a user in terms of not only what she is reading, but how much she is reading as well?

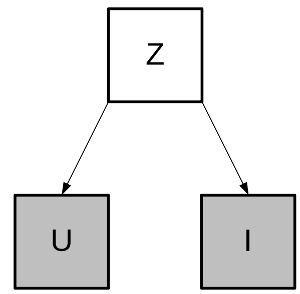
Collaborative Filtering for Changing Preferences

There could be many reasons behind a change in a user's preference. These reasons are rarely available to a recommender system. One of the advantages of collaborative filters is their lack of reliance on causes of user preferences. In the absence of data on reasons that cause user preferences to change, we divide the changes to user preferences into two groups:

1. **Systematic Changes:** These are the changes that a large number of users go through, not necessarily simultaneously, as a result of their common sequence of contexts. For example, users' life situations change; they move from being single teenagers to being married couples to being parents. Or their role in a job might change. In the context of an IT services firm a typical employee can move from being a trainee to being a software developer to being a manager. These changing contexts can change their preferences toward different types of products.

2. **Unique Changes of Individual Users:** These are also changes to users' preferences due to a random factor. For example, a rare illness in a user's family might spur her to take interest in certain types of treatments. Such an increase of interest in specific topics may not be observable in the general population. In the absence of any observable cause, these sudden changes in interest would appear random.

It is difficult to learn anything from apparently random changes without access to the underlying causes. However, it is possible to identify the systematic changes in preferences from the behavior of many users even when we do not have information about the context that could have caused the changes. The goal of this paper is to identify *common patterns of change* in user preferences. The knowledge of these patterns will allow us to predict the preference of the user in the next time period, potentially after the training data collection period, and make an appropriate recommendation for that time period.



The user (U) and the item (I) occurrences are governed by a latent class (Z). Each observation is of the form (User ID, Item ID). In this and the remaining Bayesian network representations of the models in this paper, the observed variables are represented by a filled-in square and latent variables are represented by an empty square.

Figure 2. The Bayesian Network Representing the Aspect Model of User–Item Co-occurrence

A Hidden Markov Model of User Preference

We design a model of changing user preference based on the probabilistic graphical modeling framework. It is helpful to start by examining a static model based on this framework, such as the aspect model (Figure 2). In this model the distribution over users and items is expressed as $P(U, I) = \Sigma_Z P(Z)P(U|Z)P(I|Z) = \Sigma_Z P(U)P(Z|U)P(I|Z)$, that is, the occurrence of an item in a (user, item) observation is independent of the occurrence of the user if we know the distribution over the latent class for that observation. So, if we are interested in predicting the occurrence of an item in a data record,[2] the information about the occurrence of the user in that record is only useful for predicting the occurrence of the latent variable Z, which is sufficient for computing the occurrence probability of any item I. Thus, the entire preference of a user for different items is encoded in the user's membership to the latent classes: $P(Z|U)$.

This allows us to think about a changing user preference in terms of changing membership to latent classes. A natural development from the static latent class model to a dynamic latent class model is the hidden Markov model (HMM). There are three distribution components of an HMM:

1. The *starting probability distributions* over the latent classes for each user (π).
2. The *transition probability* table between classes in adjacent time periods (A).
3. The *emission* or *observation* model that generates the data from the latent class memberships in each time period.

[2]A data records consists of one (user, item) occurrence.

In our context, the observation for each user is a sequence of visits to different blog articles in each month. The observation for each month consists of the IDs of the articles visited by the user. By modeling this process as an HMM, we make the following assertions:

1. A user's latent class memberships in a given period depend only on the user's class memberships in the previous time period (Markovian assumption). Note that we do not assume that the articles the user visits in time period t depend only on the articles the user visited in time period $t - 1$, or that if we know what the user read in one time period we have all the information to predict what the user will read in the next time period. Rather, all the observations about the user until time period $t - 1$ are taken into account to compute the user's membership in the latent classes in time period, which is used to predict the articles the user will read in time period t. This is one of the key advantages of HMM over a simple Markov model.

2. Each user can have a different starting distribution over the latent classes at $t - 1$.

3. If we know the membership of the user to different latent classes at a time period, then we have all the information needed to predict the observations of that time period.

4. The class specific observation models are global. However, a user's unique membership to different latent classes allows us to model each user's visits to blog articles in a unique way.

The starting probability and the transition probability of the proposed model have a lot of similarity with the ones pro-

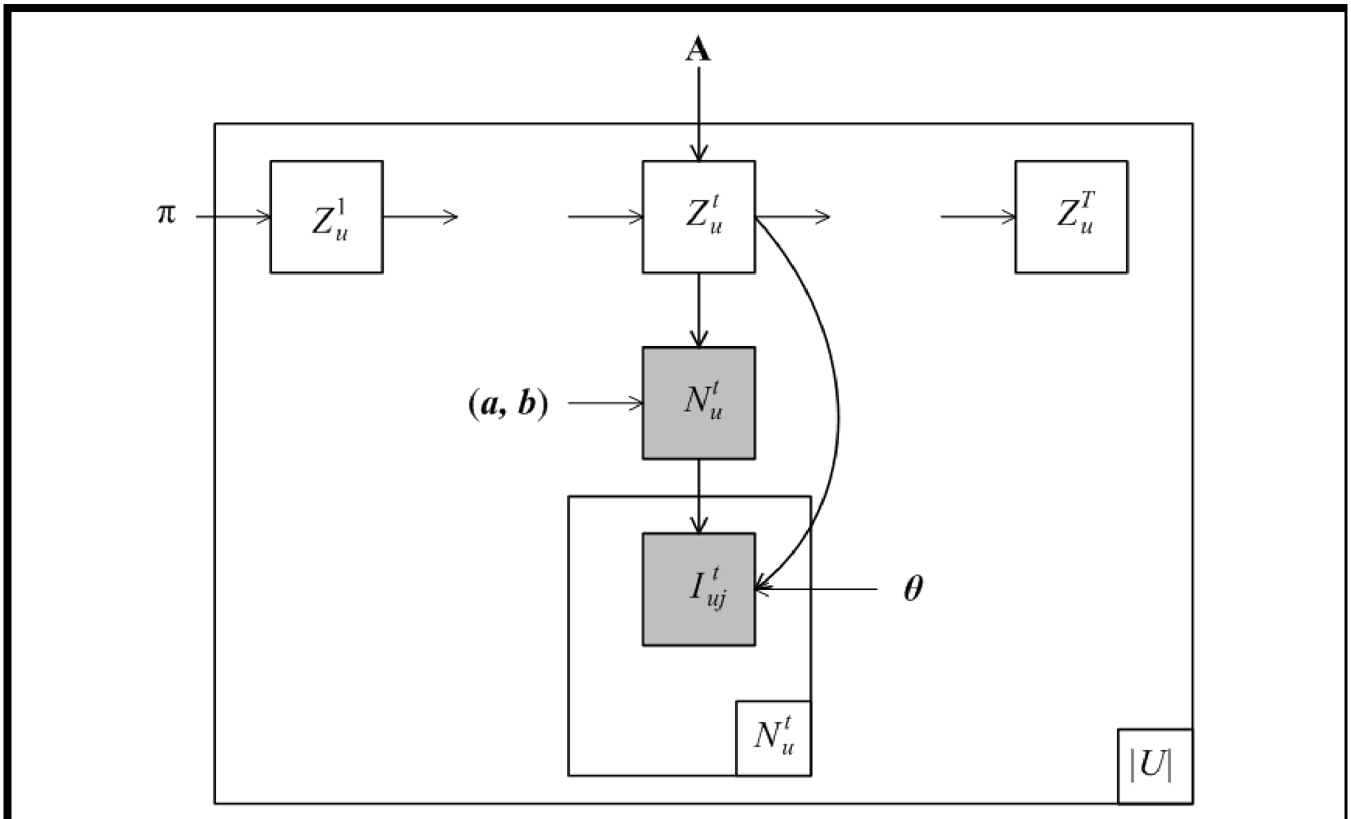


Figure 3. An HMM for the Blog Reading Behavior of the Users

posed in the HMM literature. However, the observation model is different. While in the literature one often sees one emission from the latent class for each time period, in modeling blog reading behavior of the users we observe the users reading a different number of articles in each month. Therefore, we have two distributions responsible for generating the observations in each month.

1. A distribution determining how many articles will be read in a month: N_u^t. We model this *count* as a set of class specific negative binomial distributions (NBD), each of which has a pair of parameters (a_k, b_k).

2. A distribution determining what articles will be visited in that month: I_u^t. We model this *item selection* distribution as a set of class specific multinomial distribution over all of the items. The parameters of each state specific distribution are a column of θ, denoted as θ_k. Each of the N_u^t articles is assumed to be drawn from the class specific multinomial distribution.

We use the formulation of NBD as a gamma mixture of poisson (Minka 2002):

$$\begin{aligned} P_{NBD}(N;a,b) &= \int Poisson(N;\lambda)Gamma(\lambda;a,b)\,d\lambda \\ &= \binom{a+N-1}{N}\left(\frac{b}{b+1}\right)^N\left(1-\frac{b}{b+1}\right)^a \end{aligned} \quad (1)$$

Note that, in comparison to the two emission distributions proposed here for the HMM, only the item selection distribution is estimated in the aspect model and not the number of items selected. The other point to note is that the distributions that make up the observation model are not user specific. So, their parameters only grow with the number of latent classes and not with the number of users. The proposed HMM is shown in Figure 3 using plate notation.

The variable Z_u^t is the latent class variable representing the preference of the user u at time period t. The variables N_u^t and I_u^t are the observed variables. N_u^t is the number of articles the user read in time period t. The j^{th} element of I_u^t, denoted as $I^t_{\{uj\}}$, is the ID of the j^{th} article the user u reads in time period t. $|U|$ is the number of users in the data set and $|I|$ is the number of items in the data set. The parameters of the model are described in Table 1.

Table 1. Description of Parameters

Parameter	Size	Distribution	Description
$\boldsymbol{\pi}$	$K \times 1$	Discrete distribution	Distribution of starting state of the users.
$\mathbf{A}$	$K \times K$	Discrete distribution	Each row parameterizes a distribution for a state **from** which the user will change state. Each column has the probability that a user will move **to** the state corresponding to that column.
$\boldsymbol{a}, \boldsymbol{b}$	$K \times 1$, $K \times 1$	Negative binomial	The k^{th} element of **a** and **b** are the shape and scale parameters of a gamma distribution. The NBD for the k^{th} state is a mixture of poisson distribution with this gamma mixing distribution. The NBD models the number of items selected by a user in a particular time period.
$\boldsymbol{\theta}$	$\|I\| \times K$	Multinomial distribution	Each column of $\boldsymbol{\theta}$ contains parameters of the multinomial distributions that capture preference of a class of users toward the items.

Hidden Markov Model as a Collaborative Filter

One of the building blocks of aspect model based collaborative filters is the global item selection distributions $P(I|Z)$. Part of the effort in training the model involves learning these distributions from the behavior of all the users in the system. In the proposed HMM, we learn K static global item selection distributions from the behavior of all the users in all the time periods. Thus, we retain the element of collaborative learning from collaborative filtering algorithms in the proposed HMM.

In the aspect model, in addition to the distribution over the items we also learn the unique static probabilities of each user behaving according to each of these distributions. However, in HMM for each user, a different probability distribution over the latent states in each time period is possible. As a result of this setup, a user may have moved away from a latent class representing a past preference. However, knowing the state of the user in a prior time period allows us to use the user's behavior in that time period to learn the corresponding global distribution over items. This distribution can subsequently be used to make recommendations for other users when they enter that state in the future.

Estimation and Complexity

The parameters were estimated by the expectation maximization (EM) approach (Dempster et al. 1977). In this approach, the parameters are optimized via two alternating steps:

1. **Expectation/E-step**: The distribution over the hidden variable, Z_u^t, is computed using the values of the parameters obtained so far and the observations for the user.
2. **Maximization/M-step**: The parameters are calculated such that, for a given distribution over the hidden variables, the expected log likelihood of the parameters is maximized.

It can be shown that each of these two steps monotonically increase the probability of the data (Dempster et al. 1977). At the beginning of the algorithm, the parameters can be initialized to random values, or a strategy such as the one outlined in (Bishop 2006, p. 623) could be followed.

The **expectation** step amounts to performing *inference* on the latent state variables given the observations and the current estimates of the parameters. From the Bayesian network in Figure 3 it follows that *given the parameters are known constants*, assumption of the E-step, the distribution over the states of one user is independent of the states of the other user. Therefore, for each user $P(Z_u^{1:T}|I_u^{1:T})$ can be separately calculated. $I_u^{1:T}$ is the set of items selected by the user over time periods $1 \ldots T$. The posterior distribution is a function of the parameters, although we do not write it explicitly to keep the notation clean.

$P(Z_u^t|I_u^{1:T})$ and $P(Z_u^{t-1}, Z_u^t|I_u^{1:T})$ are the summary statistics of the posterior distribution required for the M-step. The *forward–backward* algorithm is an efficient algorithm to compute these statistics from posterior distribution (Rabiner 1989). To understand how this algorithm works, we define two expressions:

$$\alpha(Z_u^t) = P(Z_u^t|I_u^{1:t})$$

and

$$\beta(Z_u^t) = \frac{P(I_u^{t+1:T}|Z_u^t)}{P(I_u^{t+1:T}|I_u^{1:t})}$$

Each can be written recursively.

$$\alpha(Z_u^t) = \left[\sum_{Z_u^{t-1}} \alpha(Z_u^{t-1}) P(Z_u^t|Z_u^{t-1})\right] \frac{P(I_u^t|Z_u^t)}{P(I_u^t|I_u^{1:t-1})}$$

$$\beta(Z_u^t) = \frac{\sum_{Z_u^{t+1}} P(Z_u^{t+1}|Z_u^t)\beta(Z_u^{t+1})P(I_u^{t+1}|Z_u^{t+1})}{P(I_u^{t+1}|I_u^{1:t})}$$

The $\alpha(Z_u^t)$'s can be computed via one forward pass over the data sequence. $P(I_u^t|I_u^{1:t-1})$ are the normalizing constants that make $\alpha(Z_u^t)$ sum to 1. They can be collected during the forward pass. The $\beta(Z_u^t)$'s are computed by one more pass going in the backward direction over the data sequence. The posterior distribution $P(Z_u^t|I_u^{1:T}, \Theta)$ is the product of the two expressions.

$$\begin{aligned}\alpha(Z_u^t)\beta(Z_u^t) &= P(Z_u^t|I_u^{1:t}) \frac{P(I_u^{t+1:T}|Z_u^t)}{P(I_u^{t+1:T}|I_u^{1:t})} \\ &= \frac{P(Z_u^t|I_u^{1:t}) P(I_u^{t+1:T}|Z_u^t, I_u^{1:t})}{P(I_u^{t+1:T}|I_u^{1:t})} \\ &= \frac{P(I_u^{t+1:T}, Z_u^t|I_u^{1:t})}{P(I_u^{t+1:T}|I_u^{1:t})} = P(Z_u^t|I_u^{1:T})\end{aligned} \tag{2}$$

Here we use the fact that $P(I_u^{t+1:T}|Z_u^t, I_u^{1:t}) = P(I_u^{t+1:T}|Z_u^t)$ because $I_u^{1:t} \perp I_u^{t+1:T}|Z_u^t$.

Using similar algebraic manipulation, it can be shown that

$$\begin{aligned}&P(Z_u^{t-1}, Z_u^t|I_u^{1:T}) \\ &= \frac{\alpha(Z_u^{t-1})P(Z_u^t|Z_u^{t-1})P(I_u^t|Z_u^t)\beta(Z_u^t)}{P(I_u^t|I_u^{1:t-1})}\end{aligned} \tag{3}$$

The complexity of the forward and backward passes increases linearly with the length of the sequence. The calculation of each $\alpha(Z_u^t)$ and $\beta(Z_u^t)$ is dominated by K^2 products. This leads to a complexity of $O(TK^2)$ for completing the expectation step.

In the **maximization** step, we maximize the expected log likelihood of the parameters, which is a lower bound on the log likelihood of the parameters (Bishop 2006)

$$\begin{aligned}&\sum_u \sum_{t=1}^{T} P(Z_u^t|I_u^{1:T}; \Theta^{old}) \log P(I_u^{1:T}, Z_u^{1:T}; \Theta) \\ &\le \log P(I; \Theta)\end{aligned} \tag{4}$$

Where I represents all of the observations, that is, all of the items selected by all of the users. Θ denotes the set of all the parameters, that is, $\mathbf{A}$, $\boldsymbol{\theta}$, $(\boldsymbol{a}, \boldsymbol{b})$. Θ^{old} represents the parameter estimates after the last iteration.

When the probability distribution over the observed and latent variables is represented as an HMM, then the log likelihood of the parameters decomposes into the sum of three components corresponding to three distributions of the HMM. The expected log likelihood is

(Initial state distribution)

$$\sum_u \sum_k P(Z_u^1 = k|X; \Theta^{old}) \log \pi_k$$

(Transition model)

$$+\sum_u \sum_{t=2}^{T} \sum_j \sum_k P(Z_u^{t-1} = j, Z_u^t = k|X; \Theta^{old}) \log A_{jk} \tag{5}$$

(Emission model)

$$+\sum_u \sum_{t=1}^{T} \sum_k P(Z_u^t = k|X; \Theta^{old}) \log P(N_u^t, \{I_{uj}^t\}|a_k, b_k, \boldsymbol{\theta}_k)$$

Note that parameters for the three models can be maximized independent of each other. Maximizing equation (5) leads to the maximum likelihood estimates of the parameters. However, MLEs run the risk of over-fitting when the size of the training data set is small and can have poor predictive power. Maximum-a-posteriori (MAP) estimates, on the other hand, are based on the assumption that the parameters are random variables drawn from a specified prior distribution. Using Bayes' theorem, the posterior distribution of the parameters can be calculated. MAP estimates of the parameters maximize the posterior distribution. By specifying the prior distribution, one can provide prior knowledge about how parameters are likely to be distributed. This reduces the risk of the estimates over-fitting to the oddities of the small training samples. When the prior distribution is conjugate to the distribution of the data, posterior distribution has the same form as the prior. This leads to tractable computation of the MAP estimates.

The distribution of a user's latent class at $t = 1$ and the distribution of the user's latent class at $t > 1$ conditioning on the user's latent class at $t - 1$ are multinomial distributions. The conjugate prior of the parameters of these distributions are Dirichlet distributions. We use the following Dirichlet prior for all of the multinomial distributions.

$$\boldsymbol{\pi}, \boldsymbol{A}_{j,:} \sim Dir(\boldsymbol{x}|\alpha_1, \ldots, \alpha_K) \tag{6}$$

where $\boldsymbol{A}_{j,:}$ is the j^{th} row of the transition probability matrix. Each α_K is set to $\frac{\alpha}{K}$. The weight of the evidence provided by each prior is α. The α was set to 100 in our experiments. The MAPs of the parameters are given by (Bishop 2006, p. 618):

$$\widehat{\pi_k} = \frac{\left[\sum_u P\left(Z_u^1 = k \middle| X; \Theta^{old}\right)\right] + \alpha_k - 1}{\left[\sum_u \sum_k P\left(Z_u^1 = k \middle| X; \Theta^{old}\right)\right] + \alpha - K} \tag{7}$$

$$\widehat{A_{jk}} = \frac{\left[\sum_u \sum_t^T P\left(Z_u^{t-1} = j, Z_u^t = k \middle| X; \Theta^{old}\right)\right] + \alpha_k - 1}{\left[\sum_u \sum_t^T \sum_l P\left(Z_u^{t-1} = j, Z_u^t = l \middle| X; \Theta^{old}\right)\right] + \alpha - K} \tag{8}$$

The emission model is novel in the context of HMMs. Each class-specific emission distribution is a NBD mixture of multinomial distributions. From Figure 3, the observations N_u^t, I_{uj}^t, and variable Z_u^t d-separates $(\boldsymbol{a}, \boldsymbol{b})$ from $\boldsymbol{\theta}$. Since in M-step the distribution over Z_u^t is fixed and the value of the observed variables are constant, the parameters of the NBD and the parameters of the multinomial become conditionally independent in the M-step. This is verified by writing out the log likelihood of the parameters for the emission distribution. The complete data, with observed and hidden variables, log likelihood decomposes as sum of functions of the two sets of parameters, and so does their expectation. Expected log likelihood of the observation model is

$$\begin{aligned}&\sum_t^T \sum_k P\left(Z_u^t = k \middle| X; \Theta^{old}\right) \log P\left(N_u^t \middle| a_k, b_k\right) \\ &+ \sum_t^T \sum_k P\left(Z_u^t = k \middle| X; \Theta^{old}\right) \log P\left(\left\{I_{uj}^t\right\} \middle| \boldsymbol{\theta}_k, N_u^t\right)\end{aligned} \tag{9}$$

Each summand can be maximized separately with respect to its parameters, which is a problem of maximizing weighted log likelihood. Maximizing the posterior probability amounts to adding log prior probability of the parameters, a function of only the parameter and not the data, to each summand and maximizing it with respect to the parameters.

Maximizing the second summand is equivalent to computing the MAP estimate of the class specific multinomial distribution over the items. There is a closed form solution for this (Bishop 2006, p. 618).

$$\theta_{ik} = \frac{\sum_u \sum_{t=1}^T P\left(Z_u^t = k \middle| X; \Theta^{old}\right) \sum_j^{|I|} 1_i(I_{uj}) + \alpha_i - 1}{\sum_u \sum_{t=1}^T P\left(Z_u^t = k \middle| X; \Theta^{old}\right) N_u^t + \alpha - K} \tag{10}$$

Where $1_i(I_{uj})$ is an indicator function that takes value 1 when I_{uj} and 0 otherwise. $\boldsymbol{\theta}_k$ is drawn from a Dirichlet prior $Dir(\boldsymbol{x}|\alpha_1, \ldots, \alpha_{|I|})$, where each $_1 = \frac{\alpha}{|I|}$. Again in our experiments α was set to 100.

There is no closed form solution for calculating the MAP estimate of the weighted NBD. We use an iterative algorithm similar to the one presented in Section 2.1 of Minka (2002) for obtaining the MLE of a NBD.

Prediction

The task of time sensitive recommender systems is to predict the articles a user will read in time period $t + 1$ given all the articles all the users have read in each time period up to t.

The estimated HMM with data observed up to can be used to compute the latent class distribution for each user in time period $t + 1$ and then compute the distribution over the observation of articles in time period $t + 1$. The probability that the item will be observed in $t + 1$ can be computed as

$$P(i \in I_u^{t+1}) = \sum_k P(Z_u^{t+1} = k)\, P(i \in I_u^{t+1}; a_k, b_k, \boldsymbol{\theta}_k) \tag{11}$$

Then the items that are most likely to be observed in period $t + 1$ can be recommended to the user. For each user u, the order of the items by equation (10) is equivalent to their order by the following quantity:

$$R(i,u) = -\sum_k P(Z_u^{t+1} = k)(1 + b_k \theta_{ki})^{-a_k} \tag{12}$$

Please see Appendix A for the derivation.

Comparison with Existing Methods

We compare the proposed dynamic model with three static algorithms and one dynamic algorithm that has been recently proposed for temporal link prediction. Each algorithm specifies a way to calculate the score for each user u and item i pair. An item i may be recommended to a user u based on this score. The score is denoted as $R(i, u)$ and formulae to calculate it are described below for each algorithm.

User–User Similarity Based Collaborative Filter

The algorithms based on the similarity between users rely on the users' prior rating on items. Since we have implicit ratings, we treat the prior visit of a user to an article as rating 1 and lack of prior visit as rating 0. This convention is often seen in the literature (Das et al. 2007). We use the framework proposed by Breese et al. (1998) to compute the scores over the items for each user.

Table 2. Summary of the HMM Algorithm for Collaborative Filtering

Algorithm	Control Parameters
1. Use data collected over time period 1 … t_{trn} for training. 2. ***Initialize*** $\boldsymbol{\pi}$, $\mathbf{A}$, ($\boldsymbol{a}, \boldsymbol{b}$), $\boldsymbol{\theta}$ to small random values. 3. ***E-step***: Compute $P(Z_u^t \mid I_u^{1:T})$ and $P(Z_u^{t-1}, Z_u^t \mid I_u^{1:T})$ using equations (2) and (3). 4. ***M-step***: *Estimate* $\boldsymbol{\pi}$, $\mathbf{A}$, $\boldsymbol{\theta}$ using equations (7), (8), and (10). Estimate ($\boldsymbol{a}, \boldsymbol{b}$) using Section 2.1 of Minka (2002). 5. If expected log likelihood has not converged, go to step 2. 6. For each user *u* compute *R*(*i, u*) of each item *i* for time period t_{trn} + 1 using equation (12). a. ***Recommendation top N*** items with highest *R*(*i, u*).	1. Set *K* to a value that maximizes the AIC score. 2. Set length of the time period to 1 month. a. Smaller if user preferences change quickly. b. Larger if constrained by computing resource.

$$R(i,u) = \overline{R_u} + \frac{1}{\sum_{v=1}^{|U|} abs(sim(u,v))} \sum_{v=1}^{|U|} sim(u,v)(R_{vi} - \overline{R_v}) \quad (13)$$

The expected rating a target user *u* would give to item *i* is computed by the sum of ratings of the other users (*v*) weighted by the similarity of those users to the target user *u*. R_{vi} is the rating given to item *i* by user *v*. $\overline{R_v}$ is the average rating of the user *v*. One of several metrics can be used for computing the similarity between two users who are represented by the vectors of ratings they have given. Some choices are cosine, correlation, inverse Euclidian distance, etc. We use correlation coefficient since it has often been used in the literature (Breese et al. 1998; Herlocker et al. 1999).

Aspect Model

The parameters of the aspect model are the conditional probability tables $P(Z)$, $P(U|Z)$, and $P(I|Z)$. They can be estimated using an expectation maximization algorithm. For a user *u*, the items can be recommended in the decreasing order of the probability of selecting a particular item:

$$R(i,u) = P(i|u) = \sum_{Z} P(Z)P(u|Z)P(i|Z) \quad (14)$$

Link Analysis

The link-analysis algorithm has been specifically developed for transactional data (Huang et al. 2007b). It represents the product selection by users as a bipartite graph. On this graph the algorithm generalizes the popular hub–authority score calculation to compute a set of product and consumer "representative" matrices. If there are *M* users and *N* products, then the product representative matrix, *PR*, is $N \times M$ and the consumer representation matrix, *CR*, is $M \times M$. Each cell of the product representative matrix contains the degree to which each product is represented by each user. Each cell of the consumer representative matrix contains the degree to which one user is represented by another. Let the user– product adjacency matrix be an $M \times N$ matrix called *A*. Then it is shown that the and matrices can be defined to be

$$PR = A^T \cdot CR \quad (15)$$

$$CR = B(A) \cdot PR + CR^0 \quad (16)$$

Where $B(A)$ is a normalized adjacency matrix such that $b_{ij} = \frac{a_{ij}}{\left(\sum_j a_{ij}\right)^{\gamma}}$. CR^0 is a diagonal matrix with weight η that assigns an additional representation score by each user to self. Before adding CR^0 to it the product $B(A) \cdot PR$ is normalized so that each column sums to 1. At the start of the algorithm, CR is set to CR^0. Iterating over equations (15) and (16) converges to the product and consumer representative matrices. Once the product representative matrix is estimated, it can be used for making recommendations. Under the link analysis algorithm, the suitability score of a product, *i*, for a user, *u*, is

$$R(i, u) = PR(i, u) \quad (17)$$

Following Huang et al. (2007n), γ was set to 0.9. In our experiments, the results were not very sensitive to the value of η. It was set to in the reported results.

Katz-CWT

Predicting a user's selection of a product can be formulated as a link prediction problem: predicting whether a link occurs between a user and a product. Many link prediction methods have been proposed in the literature. Among them the Katz method has been shown to be one of the best methods (Liben Nowell and Kleinberg 2007). This algorithm has been extended to temporal link prediction (i.e., predicting occurrence of a link at a particular time; Dunlavy et al. 2011). It has been shown to be one of the best performing algorithms for predicting occurrence of a link in a particular time period.

The Katz method computes a score indicating the potential of a future direct link between two nodes that currently do not have a direct link or edge. It is calculated for a link between and as

$$\hat{\mathbf{S}}(i,j) = \sum_{l=1}^{\infty} \beta^{l} \left|path_{i,j}{}^{(l)}\right| \tag{18}$$

Where $|path^{(l)}|$ is the number of paths of length l between node i and node j. β is a parameter that controls the extent to which longer paths are discounted.

Let A be the adjacency matrix of a bipartite graph, such as a user-item graph. Dunlavy et al. (2011) have shown that for bipartite graphs $\hat{S}$ can be approximated as

$$\hat{S} = U_K \Psi_K V_K^T \tag{19}$$

Where U_K is a matrix whose columns are the first K left singular vectors of A. V_K is a matrix whose columns are the first K right singular vectors of A. Ψ_K is a diagonal matrix with whose p^{th} element is a modified singular value of A

$$\psi_p = \frac{\beta\sigma_p}{1-\beta^2\sigma_p^2} \tag{20}$$

Where σ_p is the p^{th} singular value of A.

These scores are calculated from an adjacency matrix that is devoid of any temporal information. To convert a data set that contains time-stamped links to adjacency, it has been shown that a time discounting strategy works well (Dunlavy et al. 2011). Thus, if the data set contains links formed over T time periods the adjacency matrix is computed as

$$A = \sum_{t}^{T} (1-\theta)^{T-t} A_t \tag{21}$$

Where $\theta \in (0, 1)$is a parameter that controls how quickly older links are discounted. A_t is an adjacency matrix that contains links formed in period t. This is called the *collapsed weighted tensor* (CWT). The Katz score computed using equation (19) on an adjacency matrix constructed according to equation (21) leads to the Katz-CWT algorithm.

In our experiments, θ and β were set to best performing values for each data set after evaluating them on a validation set. The optimal value for $\beta = 0.001$, where as θ was different for each data set.

Since $\hat{S}$ contains scores indicating the potential of a link between a user and an item it can be directly used for selecting potential items to recommend. The recommendation score of the item i for user u is

$$R(i,u) = \hat{S}(u,i) \tag{22}$$

Recommending Popular Items

This is a popular non-personalized recommendation strategy being used by many prominent online retailers in conjunction with personalized recommender systems. It is also a popular choice for baseline performance in the collaborative filtering literature (Rashid et al. 2002). In this strategy, the recommendation score of the items is calculated as

$$R(i,u) = \sum_{v=1}^{u} R_{vi} \tag{23}$$

In addition to these five algorithms, a recently proposed dynamic collaborative filter for explicit rating data, known as timeSVD++, was extended to the implicit rating scenario. Although timeSVD++ has been very successful on explicit rating data, its extension to the implicit rating scenario by treating each selection as a rating 1 and lack of selection as a rating 0 was ineffective for making recommendations. Therefore, it is excluded from the set of reported comparisons.

Experiments with Real World Data

Each algorithm is trained on data up to a certain time period t. The trained algorithm is then used to predict what each user will select in period t + 1. Using the convention in the literature, the data set was limited to only those users who have read at least a certain number of articles and only those articles that have been read at least a certain number of times. First we present the performances of the algorithms on blog reading data by setting both these thresholds at 400. Later we present the results of sensitivity analysis where the threshold is varied as well as the results on two other data sets.

Model Selection

The optimal number of latent classes is determined using AIC criterion.[3] We find that using 5 latent classes for the static model and 25 latent classes for the HMM model are optimal. The relatively higher number of latent classes that the dynamic model requires has practical implications. The complexity of the algorithm grows as a square of the number of classes. Despite this, as the complexity of the EM algorithm grows only linearly with the size of the data set, we are able to complete the task relatively quickly (within 10 to 20 minutes using commodity hardware).

Recommendation Performance

The performances of the six recommender systems are measured by their *precision* and *recall* scores. Each algorithm is used to calculate the recommendation score of all articles for a user. Then the top 5 or top 10 highest scoring articles are recommended for the user. Only the articles that the user is observed to visit in time period $t + 1$, the test set, are considered the correct recommendations. Precision, P, of the algorithm is the *fraction of the recommended set* that is correct. Recall, R, is the *fraction of the correct articles* that is recommended. If more items are recommended the precision will decrease, but recall will increase. The harmonic mean, F, of the precision and recall is often used to summarize the two numbers (Herlocker et al. 2004).

$$\frac{1}{F} = \frac{1}{2}\left(\frac{1}{P} + \frac{1}{R}\right) \tag{24}$$

The dynamic model requires sequences of adequate length to learn the transition probabilities. We use data collected over time period $1 \ldots t$ to train the algorithms, where $t = 15 \ldots 21$ (i.e., we make sure that at least two thirds of the data is available for training). The test set consists of the articles each user visited at time period $t + 1$. The precision, recall, and their harmonic means of each algorithm are computed for each train–test set and averaged. We find that the HMM based dynamic model often performs as well as the best among the alternative algorithms we evaluated, and sometimes has an advantage over them that is statistically significant. Among the algorithms to which it is compared, the Katz-CWT algorithm comes closest.

The aspect model is close to the proposed HMM with one key difference. In the HMM users are allowed to change their preferences from one time period to next, whereas in the aspect model they are not. Therefore, the improvement in the performance of the HMM over the aspect model can be attributed to the explicit modeling of changes in user preferences.

For each train–test split, each algorithm produces an ordered list of items. Precision and recall measures at the top-5 or top-10 level examine the recommendation quality of the algorithms if we are to recommend only the first 5 or first 10 of the items in this ordered list. However, if we are interested in the quality of the algorithms over the entire ordered list, then a *receiver operating characteristic* (ROC) curve is an intuitive way to compare multiple algorithms (Swets 1963).

To draw an ROC curve, one recommends items from the top of the ordered list while comparing the recommended items with the correct list of items that should be recommended. Two quantities are calculated in the process: the fraction of incorrect items recommended (false positive rate or FP) and the fraction of correct items that are recommended (true positive rate or TP). Thus, for each item in the list we generate a pair of numbers (FP, TP), used as the X and Y coordinates, respectively, to draw a curve. It is easy to note that when there are no items recommended all of the methods will produce (FP = 0, TP = 0). As more items are recommended both of these numbers monotonically increase. When all of the items are recommended from the list every algorithm would have (FP = 1, TP = 1). A perfect recommender system would retrieve all of the relevant items before retrieving any non-relevant items (i.e., it would obtain a true positive rate of 1 while having a false positive rate of 0). Only after this, retrieving any more items would increase the false positive rate. Therefore, a perfect recommender system would have the highest possible ROC curve. When comparing two algorithms, the one with a higher ROC curve is a better performing algorithm. A convenient, albeit less informative, summary of the ROC curve of two algorithms is the area-under-the-curve (AUC). The algorithm with higher AUC is the better performing algorithm.

For each algorithm, we compute the ROC curves for each user and each train–test split. We calculate the average ROC curve for each algorithm by following the *vertical averaging* strategy proposed by Macskassy and Provost (2004), where

[3]We find that the BIC criterion penalizes the models too aggressively for their complexity and suggests a very small number of classes (< 5) as optimal. Such a small number of classes does not lead to the best performance.

Table 3. Precision (*P*), Recall (*R*), and F-Scores of the Six Algorithms

Length of the time period = 30 days	Top 5			Top 10		
	P	***R***	***F***	***P***	***R***	***F***
Dynamic Model	**0.0667**	**0.1748****	**0.0956***	**0.0552**	**0.2665***	**0.0907**
Aspect Model	0.0368	0.0631	0.0460	0.0343	0.1115	0.0519
User–User Similarity	0.0409	0.0796	0.0534	0.0376	0.1381	0.0588
Popular	0.0194	0.0407	0.0262	0.0164	0.0678	0.0263
Link Analysis	0.0351	0.0692	0.0462	0.0292	0.1041	0.0453
Katz-CWT	0.0645	0.1518	0.0900	0.0514	0.2262	0.0835

Notes: Following this example of statistical comparsion of performances of multiple algorithms in the literature (Huang et al. 2007a), we perform a parid t-test between the scores of the top-2 algorithms. The cells with an asterisk (*) have an advantage that is statistically significant at the 0.10 level. The cells with a double asterisk (**) have an advantage that is statistically significant at the 0.05 level.

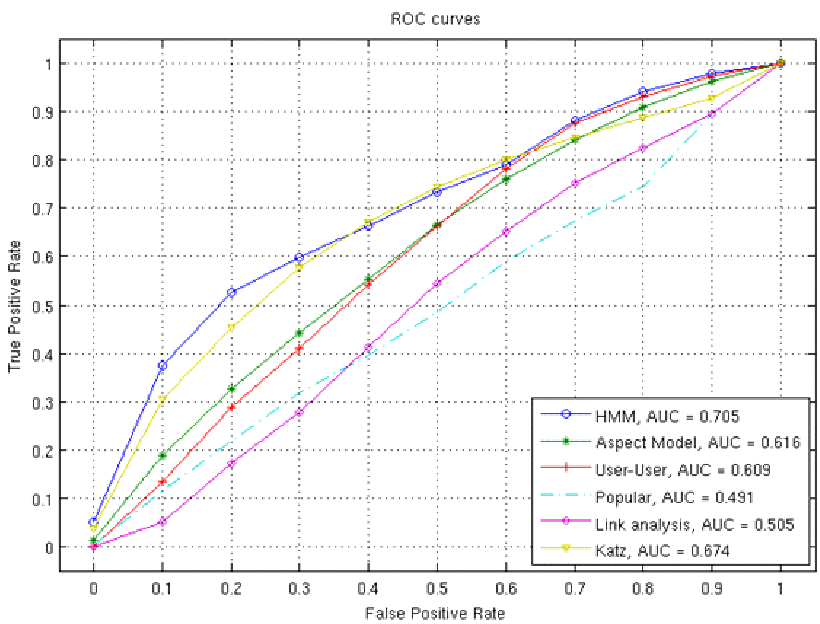


Table 4. Average ROC Curves and AUC Values of Each Algorithm

the true positive rates are extracted from each ROC curve at predetermined false positive rates and averaged. The AUCs for all of the ROCs of an algorithm are also averaged to arrive at the average AUC for the algorithm. The average ROCs and the AUCs are shown in Figure 4.

It is evident that the HMM outperforms the static models. Often the quality of interest is the performance of the algorithms at the top of the recommendation list (Järvelin and Kekäläinen 2002). As we can see in this portion of the recommended list, which translates to the initial half of the ROC curves, the HMM outperforms the static models by a significant margin. The confidence bands of the curves are also calculated following Macskassy and Provost. In the first half of the graph, the difference between the ROC curve for the HMM and the other static model is significant at the 95 percent level. However, the confidence bands are omitted from Figure 4 for the sake of clarity.

Sensitivity Analyses

The results shown thus far are obtained from a subset of data that contains only those users who have read more than 400 blog articles and only those blog articles that have been read by more than 400 users. These thresholds affect the density of the user–article matrix. In addition, all results were obtained by using 25 latent classes in the HMM and 5 latent classes for the aspect model. These were the models that produced the best results for each method. In a subsequent set of experiments, we varied the thresholds of the user and item selection. For each threshold we evaluated the methods over a range of latent classes. We found that the performance of the HMM generally improves with the number of latent classes and with the density of the data (Appendix B, Appendix C).

In the first set of experiments the length of each time period was fixed at 1 month. This seems to be a reasonable choice because the firm uses a non-personalized method that recommends to everyone the most popular articles of the previous month. This suggests that for blog article recommendation, the firm considers it appropriate to generate new recommendations each month to keep up with the changes in the blogosphere and user interests. In a second set of experiments, we vary the length of the time period for the dynamic model. The advantages of the dynamic model are more pronounced when the length of the time period is shorter (e.g., 1 week). However, when the length of the time period is much longer (e.g., 2 months), the dynamic model does not have an advantage (see Figure 5).

There are several factors that would guide one's choice of time period length. Users' interest in different products might change at different rates. Therefore, we should use time units that best capture the changes in users' interests. For example, if we are interested in tracking users' interests in news or blog articles we might want to use a shorter period than if we are interested in tracking users' interests in movies. The reason is that the interest in movies might change slowly over time as a function of the person's age, whereas the interest in certain news topics might last only a few days. In such a situation, if we use time periods that are too long, we might miss any change in the users' behavior within that time period.

The other factor to consider is that longer time periods would lead to fewer sequences from which to learn the state switching behavior. This would reduce the reliability of the learned transition probability matrix and can hurt the performance when the time periods are coarse. However, the complexity of the dynamic algorithm grows linearly with the length of training sequence. Therefore, using shorter time periods would require a longer training period for the dynamic algorithm.

Evaluation on the Netflix Prize Data Set

Netflix has made available a data set containing over 100 million ratings, containing 17,770 movies and approximately 480,000 users (Bennett and Lanning 2007). The data set consists of users' ratings on movies along with the timestamp of the rating. Using this data set, we predict which movies a target user will rate in a given test period. To the extent that users rate all the movies they watch, predicting which movies the user will rate is equivalent to predicting which movies they will watch.

In the first set of evaluations, we use data from users who have rated at least 2,000 movies. This results in a data set with 1,212 users and 5,264 movies. The set of algorithms described in the earlier subsection "Estimation and Complexity" are evaluated on this resulting data set. The number of states used for the HMM and the aspect model were decided using AIC criteria—as was done in the previous section. The parameters of Katz-CWT were set based on the performance on validation data set.

The precision and recall values at the top 5 and top 10 levels are shown in Table 4. The comparison is not affected if we increase the sparsity of the data. However, two of the algorithms (user–user similarity based collaborative filter and the link analysis method) could not be completed because of their memory requirements. To keep the data set size manageable, we randomly sampled 30,000 users to use in our experiments.

We can see an advantage of the dynamic algorithm over the static algorithms. The best among the algorithms compared to the proposed HMM-based algorithm is the Katz-CWT algorithm. The HMM outperforms the Katz-CWT when the data set is dense. We suspect the advantage of the HMM over Katz-CWT is because the Katz-CWT method relies on a time discounting strategy that does not use the old data as effectively as the HMM does. This behavior of the Katz-CWT algorithm is further explored in the following section, "Simulation of Changing Preferences," with the help of a simulation study. However, we find that when the data set is sparse, the difference between the HMM and the Katz-CWT narrows to a level that is statistically insignificant.

Evaluation on a Last.fm Data Set

Last.fm is an Internet-based personalized radio station and music recommender system. When the users of the service listen to music through a supported music player, last.fm collects data on their music listening behavior. This data is used by last.fm to make personalized music recommendation

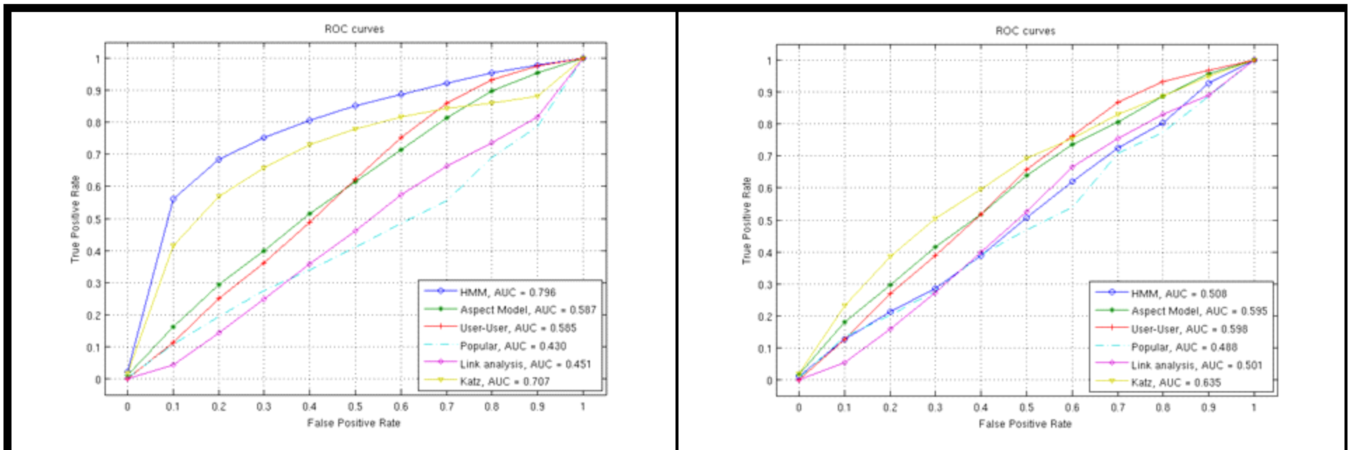


Table 5. ROC Curves with Length of the Time Period Set to 1 Week and to 2 Months

Table 4. The Precision and Recall on the Netflix Prize Data Set for New Movie Recommendations

Algorithms	**Top 5**			**Top 10**		
	P	***R***	***F***	***P***	***R***	***F***
User and movies occurring > 2,000 times included			1,212 users and 5,264 items			
Dynamic Model	**0.1621**	**0.0310**	**0.0503***	**0.1452**	**0.0529***	**0.0748****
Aspect Model	0.0652	0.0052	0.0093	0.0636	0.0100	0.0166
User–User Similarity	0.0798	0.0067	0.0121	0.0762	0.0125	0.0210
Popular	0.0770	0.0061	0.0110	0.0731	0.0111	0.0189
Link analysis	0.0780	0.0062	0.0113	0.0742	0.0114	0.0194
Katz-CWT	0.1563	0.0281	0.0458	0.1397	0.0465	0.0665
User and movies occurring > 500 times included			30,000 users and 9,284 items			
Dynamic Model	**0.1184****	**0.0344***	**0.0523****	0.1045**	0.0587	0.0738**
Aspect Model	0.0571	0.0072	0.0126	0.0543	0.0137	0.0216
Popular	0.0645	0.0081	0.0142	0.0612	0.0152	0.0240
Katz-CWT	0.0894	0.0311	0.0452	0.0805	0.0543	0.0634
User and movies occurring > 20 times selected			30,000 users and 17,753 items			
Dynamic Model	**0.0731****	0.0334	**0.0457**	**0.0663****	0.0606	**0.0631**
Aspect Model	0.0362	0.0078	0.0128	0.0344	0.0151	0.0209
Popular	0.0392	0.0082	0.0134	0.0374	0.0158	0.0221
Katz-CWT	0.0622	**0.0360**	0.0454	0.0559	**0.0624**	0.0587

Notes: Users and items that occur at least 1,000 times were included. User–user similarity based collaborative filter and the link analysis method could not be completed on this subset of the data due to their memory requirement. We perform a paired t-test between the scores of the top-2 algorithms. The cells with an asterisk (*) have an advantage that is statistically significant at the 0.10 level. The cells with a double asterisk (**) have an advantage that is statistically significant at the 0.05 level.

Table 5. The Precision and Recall Scores on the last.fm Data Set for Artists Recommendations Using Two Sparsity Levels of Data

Algorithms	Top 5			Top 10		
	P	*R*	*F*	*P*	*R*	*F*
Artists listened to by > 20 users included				978 users and 7,150 artists		
Dynamic Model (K = 20)	0.0389	0.0135	0.0200	0.0340	0.0245	**0.0285**
Aspect Model	0.0282	0.0080	0.0122	0.0259	0.0152	0.0186
User–User Similarity	0.0409	0.0137	**0.0201**	0.0361	0.0240	0.0283
Popular	0.0292	0.0097	0.0140	0.0262	0.0168	0.0199
Link Analysis	0.0374	0.0124	0.0182	0.0329	0.0209	0.0250
Katz-CWT	0.0371	0.0134	0.0189	0.0329	0.0229	0.0263
Artists listened to by > 100 users included				92 users and 1,342 artists		
Dynamic Model (K = 30)	0.0473	0.0314	**0.0369**	0.0423	0.0525	0.0469
Aspect Model	0.0308	0.0172	0.0217	0.0282	0.0310	0.0290
User–User Similarity	0.0482	0.0297	0.0357	0.0434	0.0526	**0.0470**
Popular	0.0326	0.0190	0.0234	0.0295	0.0326	0.0303
Link Analysis	0.0424	0.0247	0.0307	0.0374	0.0436	0.0397
Katz-CWT	0.0437	0.0289	0.0337	0.0393	0.0501	0.0430

Note: The parameters of Katz-CWT were set based on the performance on validation data set. The number of states of the dynamic model was determined for the two data sets using AIC criterion. A t-test to compare the HMM and the user–user algorithm reveals that the difference in their performances is not statistically significant.

at their online radio station. A part of this data has been collected and made available by Òscar Celma[4] with permission from last.fm[5] (Celma 2010). This data set contains time stamped records of users' music listening activity. It has 992 users who listed to a total of 177,000 artists. The data spans 53 months.

The algorithms were evaluated on this data set on the task of predicting the artists a user will listen to in a particular time period. To keep the evaluations of all three data sets consistent, only the new artists the users listened to in the test period were used for evaluation. The length of the time period was set to 1 month. The precision and recall scores at top 5 and top 10 levels are shown in Table 5.

On this data set, the simple user–user similarity-based collaborative filtering algorithm performs as well as the proposed HMM-based algorithm. In addition, the gaps between the dynamic algorithm and the other algorithms are narrower than they were on the blog reading data set and the movie watching data set. The comparisons do not change much when we increase or decrease the sparsity of the data. To understand why this is the case, note that the music listening data has a different characteristic than the previous two data sets. People often listen to music they like multiple times. This is recorded in the last.fm data set. They are less likely to repeatedly view movies or read blog articles. In addition, as the last.fm recommender system learns about a user's taste in music it is likely to recommend to the users more and more music that is created by the artists they like. This increases the homogeneity in the artists the users select. Thus the change in users' music listening could be less than the change in their movie watching or blog reading. This becomes apparent when we examine the state transition matrices of the HMMs learned from the three different data sets. The average probability of a user leaving a state (i.e., average of the off diagonal elements of the transition probability matrix) at the default experiment settings are 0.87, 0.59, and 0.18 for the blog reading data set, Netflix data set, and last.fm data set respectively. This suggests that user preferences are changing most in the blog reading data set and changing least in the music listening data set.

Simulation of Changing Preferences

Further insight into the performance of the algorithms is obtained by analyses of simulated data sets generated from changing preferences of different types. The data sets have

[4]http://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/lastfm-1K.html.

[5]http://www.last.fm/.

the following attributes: There are N_u users, K different preference states, and N_i distinct items that users in a particular state prefer. So, there are $K \times N_i$ distinct items in the simulated data set. Each user is observed over T time periods. In the first time period, each user starts at a state randomly chosen from the K states. The states are numbered from $1 \ldots K$. In each time period the user moves to a state according to a transition model described below. When a user assumes a given state, the user prefers a particular set of N_i items over the other $(K - 1) \times N_i$ items. This is modeled via a state-specific multinomial distribution over the items. This distribution contains a higher probability of selecting each of the N_i preferred items and lower probability of selecting the other items. Each user selects N_{obs} items in each time period.

The algorithms presented in the previous section are evaluated on this simulated data set as described in that section.

Transition Models

To understand the behavior of different algorithms when different types of dynamics are present in the user preferences, we implement three different types of state transitions.

Random Walk among States with Different Step Sizes

In this transition model, user is simulated to draw a random number from a normal distribution . The state of the user in the next time period is obtained by adding this number to the current state and rounding the result to the nearest integer between 1 and K. The advantage of this strategy is that by changing the standard deviation, σ, of the normal distribution one can control the amount of change of the users' preference. For small values of σ, the users will stay in or move to a state close to their current states. When σ is large, the users are likely to explore further away from their current state.

In Figure 6, the results of the six algorithms are reported for the following values of the simulation parameters: $N_u = 50$, $K = 10$, $N_i = 50$, $T = 50$, $N_{obs} = 50$, and $\sigma = \{0, 0.25, 0.5, 1\}$.

The left subplots of Figure 6 show the empirical state transition probability matrix computed from the simulated states. The random walk strategy results in empirical transition probability matrices that are centered on a diagonal matrix. This might seem more restrictive than it actually is. The state numbers could be shuffled to obtain a more random-looking transition probability matrix without affecting the dynamics of the model.

The right subplots show the ROC curves of the six algorithms. When the model is static (diagonal transition probability matrix), some of the static models, such as link analysis and the user–user similarity based model, perform as well as the HMM. However, as we introduce dynamics into the user preferences, the HMM begins to outperform the other models. When the user preference changes a lot, the Katz-CWT algorithm is the second best performing algorithm. It outperforms other algorithms that do not consider the temporal nature of the preferences at all.

In this setup, performances of all the algorithms, even the HMM that is designed to handle changing preferences, suffer when user preferences change. This is because even when an HMM obtains the transition probability matrix, similar to the one in the bottom row of Figure 6, there is inherent uncertainty regarding which state the user will go to in the next time period. HMM is able to put most of the probability mass over a few of the states, but it is not able to predict every time the state the user will be in. This is a harder scenario than the transition behavior shown in the top row of Figure 6, where HMM and other algorithms more accurately know which state the user is in and will be in at the test time period.

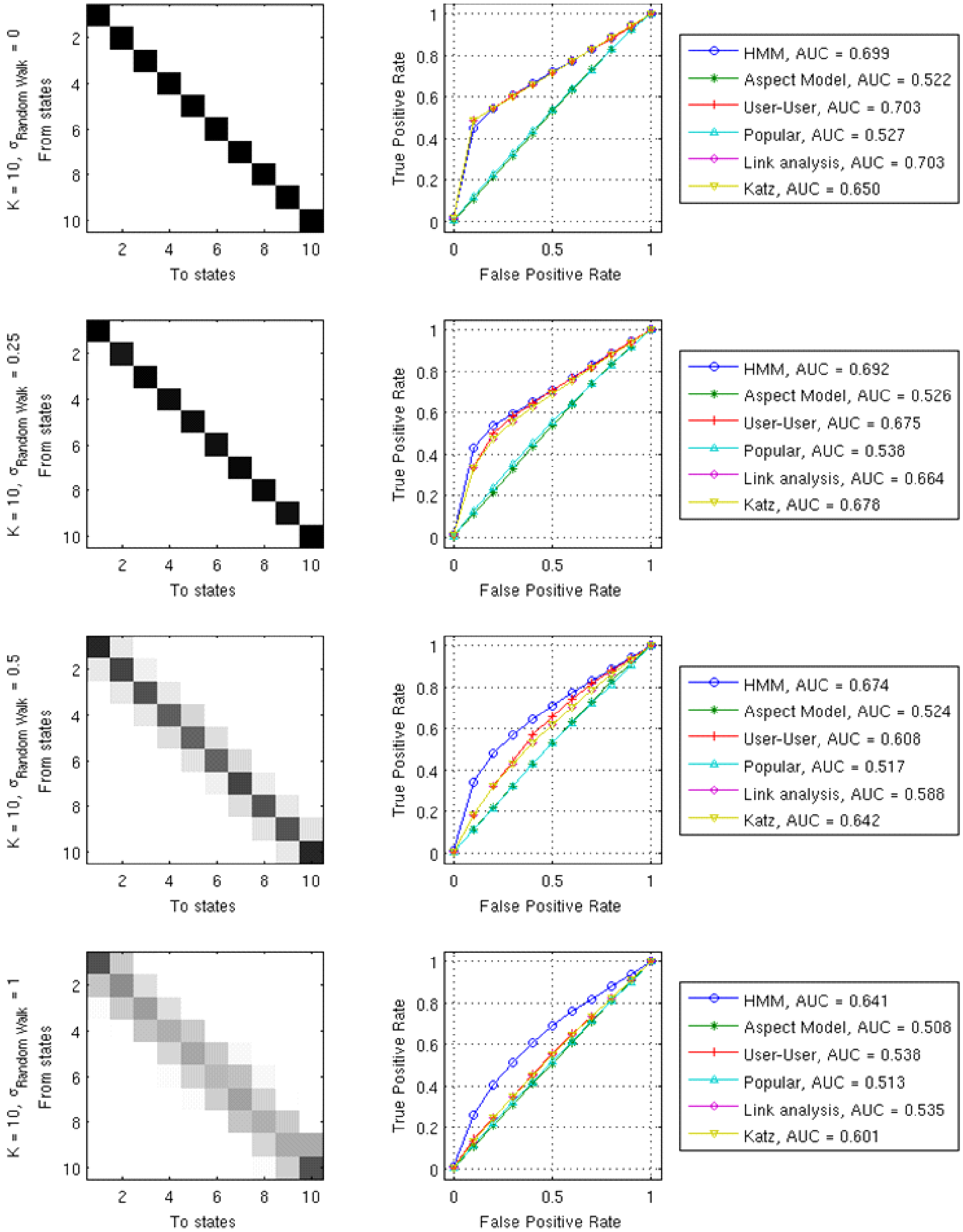
State Changes Predominantly in One Direction

In this transition model, a user moves to the next state with a high probability p_H and all other states with a low probability p_L. This models the phenomenon where, over time, users grow out of a particular type of product and move to the next level of product more or less permanently (i.e., the probability of them revisiting a state is low). Since there is a finite number of states, after a certain number of time periods the users will remain in a final state from which they do not move. Once the users are in the final state, their behavior mimics that of a user whose preference does not change.

As the p_H increases, users will converge to the final state quicker, because from each state they move to the next state with a higher probability. In addition, when the number of K states is small, users will reach the final state quickly. The gap between static algorithms and dynamic algorithms should be narrower when users reach the final stationary state quickly. The gap should be wider when they travel over more states and take longer to reach the final state. This is found and illustrated in Figure 7.

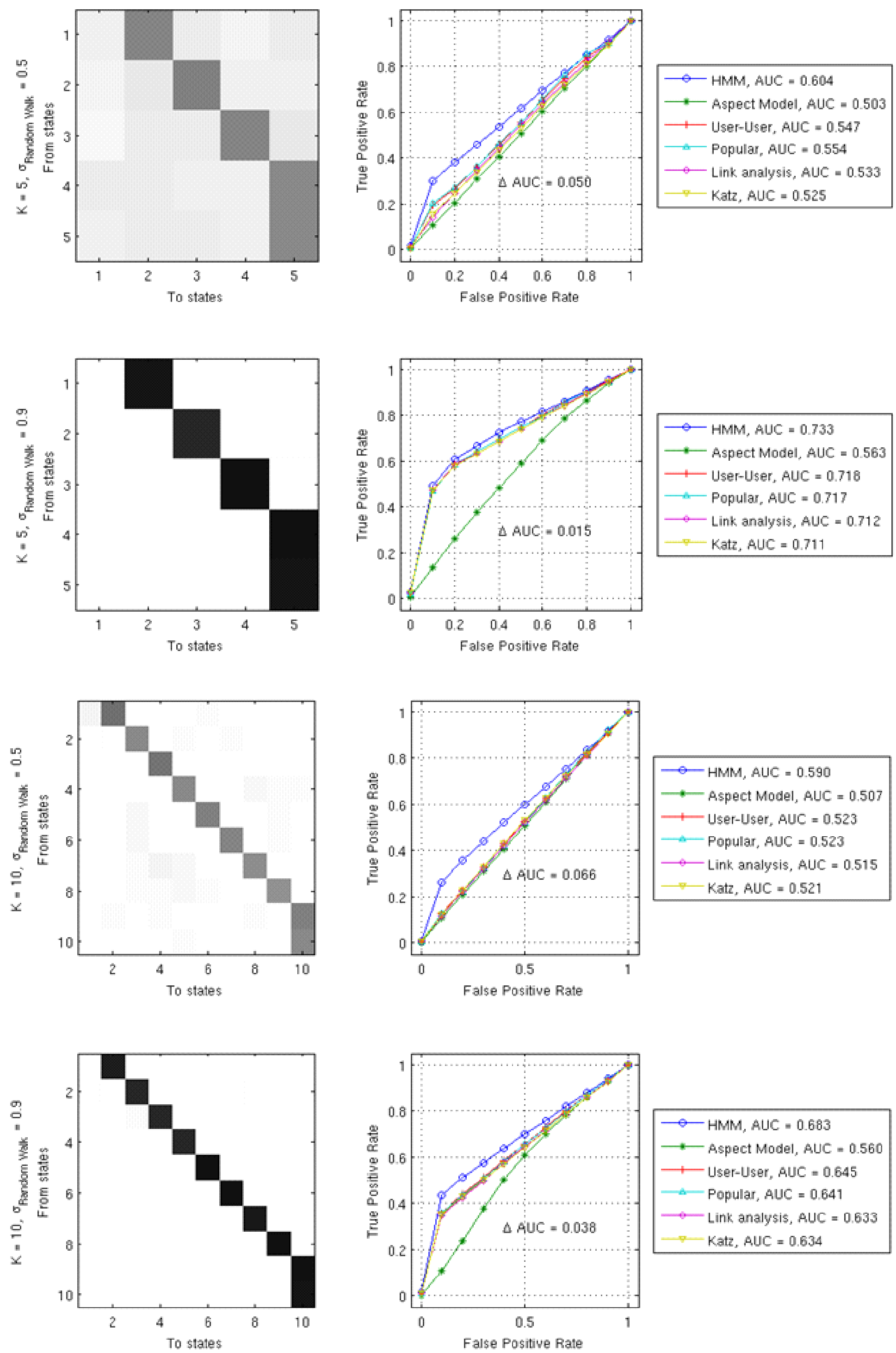
Repeating State Changes

Using the insight obtained so far it is easy to create a transition probability matrix that has such dynamic behavior that the static models perform rather poorly. One example is



Notes: The left plot in each row shows the transition probability matrix and the right plot shows the ROC curves of the compared algorithms. In the first row, the users do not change their states. In this case, some of the static models methods perform as well as the proposed dynamic model. However, as soon as we introduce some dynamics (i.e., users start switching their preference states), we start to observe the advantage of the dynamic model. This advantage grows as the amount of dynamics increases.

Figure 6. Effect of Users Changing Their Preference According to a Random Walk with Varying Step Sizes



Notes: When the user takes longer to converge to the final stationary state, the advantage of the dynamic model is larger. The users can take longer to converge to the final stationary state either due to a lower probability of moving to the next state or due to a larger number of states. *ΔAUC* is the difference in performance of the HMM and the best among the other algorithms.

Figure 7. Performances of the Algorithms When the State Changes are Unidirectional

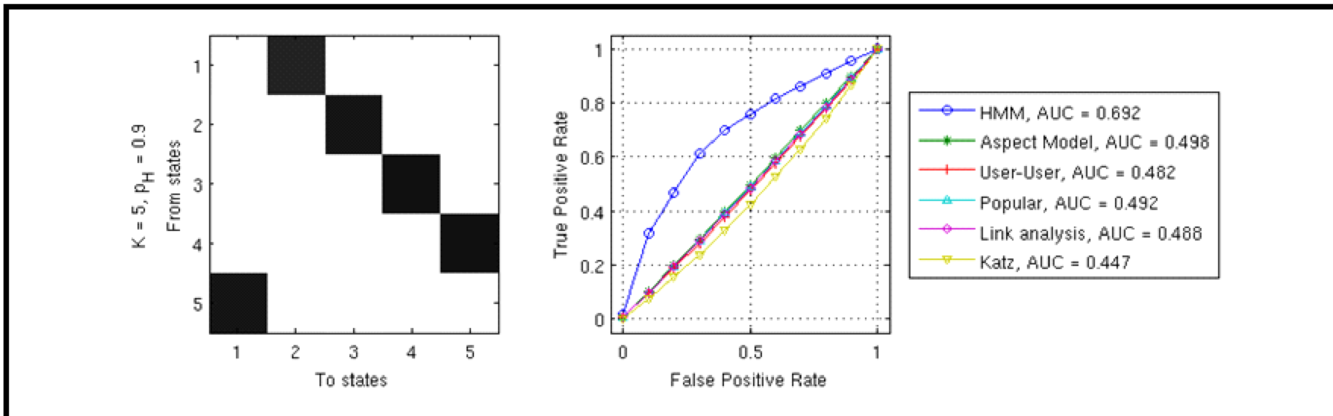


Figure 8. Static Algorithms Perform Poorly When Preferences Change Cyclically

shown in Figure 8, where users cycle through the states. As this set of transitions does not allow the user to stay in any one state for long, the static algorithms perform poorly.

Note that in this case the Katz-CWT algorithm, which applies a decaying weight to the older data, performs worse than other algorithms. To understand this better, we calculate the scores each algorithm assigns to the items for a set of test periods (Figure 9). Consider the last period in Figure 9 corresponding to T = 72. Here we see how the reader selects the items to read: she picks heavily from the first 100 items (some items multiple times) and sparsely from the rest of the 400 items. The HMM plot shows that it is able to approximate this pattern by placing the reader in a state where the weights given to the items closely mimic the observed behavior. The competing models fail to discern this pattern.

The data relevant for making a recommendation occurred time periods ago. By discounting this data heavily, the Katz-CWT algorithm performs worse than other static algorithms. In Figure 9, we can see that the Katz algorithm assigns a lower score to the relevant items that occurred K time periods ago than to the other less relevant items that were selected by the user more recently. On the other hand, HMM is able to assign a more appropriate weight to the items. This is because HMM infers the state in which the data in each time period was generated (in the E-step) and is able to selectively use the data to estimate the parameters of the generating state.

The static models project a much diffused score over all of the items for each of the test periods. This is because static models fit one model to all the data generated by a user in all time periods in the training data and use the same model to make prediction for the test period. On the other hand, the HMM creates a much more focused distribution over the items for each test period. This is because it predicts the state of the user in the test period and estimates the user's preference score toward the items when the user is in the predicted state.

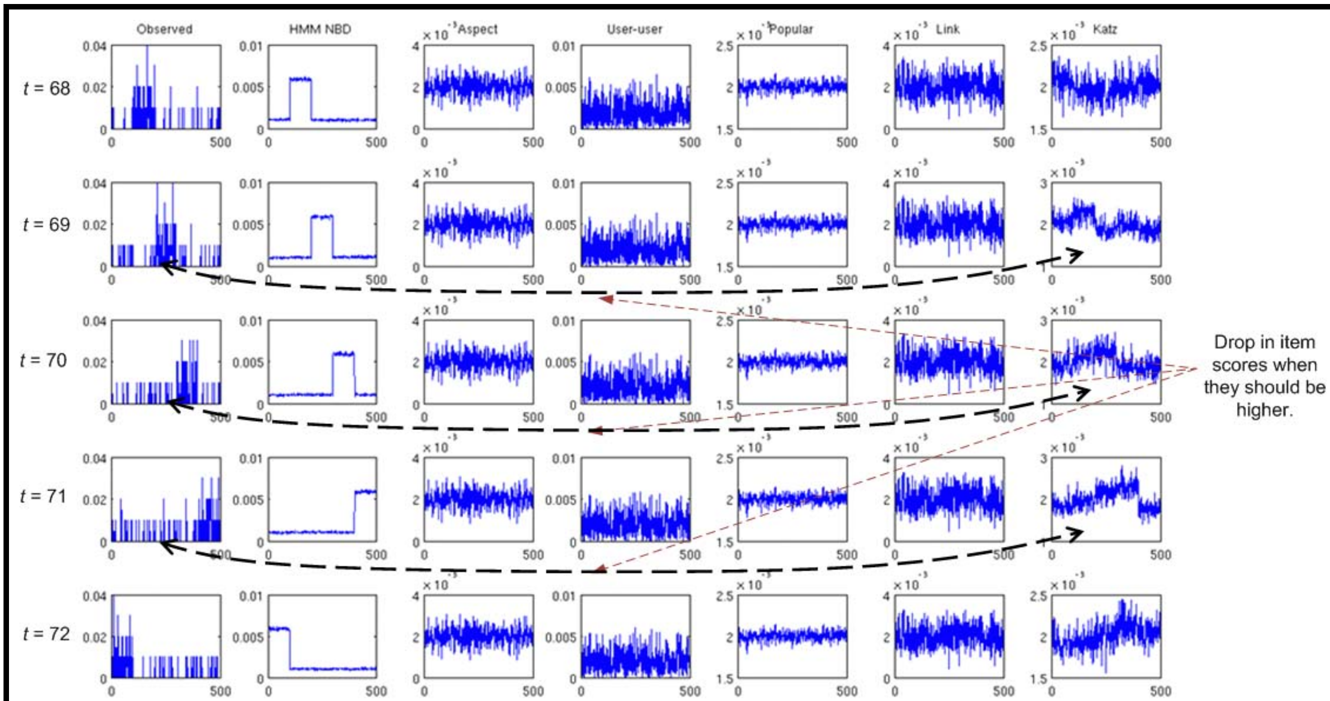
Conclusion

Summary

We present a hidden Markov model for collaborative filtering that accounts for changing user preferences. The presented algorithm is designed for implicit ratings or transactional data. Despite evidence from the literature that a user's preference can change over time, there has been very little work in the collaborative filtering literature that attempts to account for this phenomenon. We present one of the first attempts to fill this gap.

There are several challenges in recommending items to a user when the users' preferences are changing. These include the challenge that the user preferences might be different during the test period than they were in the training period, uncertainty about which of many possible preferences of a user generated the data at any time period, etc. In addition one usually does not observe multiple ratings from a user on an item. Therefore, a collaborative filter must learn changes in a user's preference from her rating on distinct items over time.

We propose an HMM to address these challenges. The preference of each user is represented as degree of membership in a set of latent classes. Each latent class represents a global preference pattern that governs the number of items selected in each month and the selection of those items. This dual purpose of the latent classes requires a novel observation model



Notes: The data set was generated using $T = 100$, $N_u = 100$, $N_i = 100$, $N_{obs} = 100$, and $K = 5$. The transition probability matrix is described in Figure 8. The five sets of scores are obtained for the first five time periods where the algorithms were evaluated: 68, 69, 70, 71, and 72. Data generated prior to each of these periods are used to train the algorithms for the corresponding test period. There is a drop in scores by the Katz-CWT algorithm for the items that should be scored high. Due to the cyclical nature of state transition, the relevant data for the test period occurs K time periods prior to the test period. The discounting strategy used in the Katz-CWT algorithm gives more weight to the recent data than older data. This raises the score for the products selected by the user in the recent time periods while reducing weights for the products selected by the user K periods ago that would have been relevant for the test period.

Figure 9. The Scores Assigned by Various Algorithms to the Items for One Randomly Selected User Over Five Consecutive Time Periods

of the HMM. We model the observations as a negative binomial mixture of multinomial distribution. We also use an estimation procedure based on the *forward–backward* algorithm that scales to large data sets.

The proposed algorithm is evaluated on three real-world data sets. The first data set is collected from a large IT services firm over 22 months. The data set consists of time stamped records of employees' visits to blog articles on the corporate blog network. The second data set is the Netflix prize data. The time stamped events of users' rating of movies are used in our study. The rating values are ignored. The third data set contains the music listening history of users of last.fm online radio. The proposed algorithm is compared with five other algorithms. They include the user–user similarity-based collaborative filter, link-analysis method for recommending using transactional data, and the Katz-CWT method for temporal link prediction applied to user–item data.

We find that the proposed HMM-based algorithm performs as well as the best of the algorithms we evaluated on the blog reading data set and on the Netflix prize data set. In some specific conditions of the data set, when the sparsity is low, we find that the HMM outperforms the other algorithms on these two data sets. On the last.fm data set, the user–user similarity-based collaborative filter performs as well as the HMM-based algorithm. The strong performance of the user–user similarity-based static collaborative filtering algorithm is traced to the static nature of the music listening data set.

Upon examining the methods using time units of different lengths, we find that when the time units are coarse (e.g., several months), the performance of the dynamic algorithm suffers. However, when the time units are shorter (e.g., a month or less), the dynamic algorithm outperforms the static algorithm. The improved performance of the algorithm with shorter time periods is achieved at a higher computational cost. The time to complete one EM iteration of the HMM increases linearly with the length of the sequences. We expect that the optimal length of the time unit will depend on the type of product to be recommended. For the products for which the preference of a user can change quickly, one would need to use a shorter time period to capture any change in user preference. A more detailed examination of the effect of the time unit on algorithm performance for different types of products is left as a topic for future research.

The properties of the algorithms are further examined by conducting a simulation study. Different degrees of dynamism in the user preference are simulated and the resulting data is used to evaluate the algorithms. It is observed that when the user preferences are static, the proposed HMM-based algorithm performs as well as the static algorithms. As we make them less static and increase the rate of change of the users' preferences, the advantage of the dynamic model increases. In addition, if the user exhibits a repeating pattern of change, the performances of the static models can be much worse than the proposed dynamic model. We also observe that the HMM does a much better job of tracking the users' changing preferences through the test period than the static models do.

Implications

Due to the information overload faced by the users of modern information systems, users and firms are increasingly relying on information filtering systems such as collaborative filters. There are several classes of products that are consumed by users repeatedly over a long period of time (e.g., movies, music, news stories, etc.). Evidence from the literature and our own examination of blog reading data suggests that user preference changes over time. This work shows that by taking into account the changes in the user's preferences one can make more effective recommendations when (1) the data is generating from changing user preferences and (2) adequate training data is available.

Training the dynamic model takes more computing resources than training the static models we compared. Therefore, this should probably be done at off peak hours. However, generating the recommendation for individual users using HMM is just as quick as other model-based collaborative filters. Therefore, it is suitable for use, for example, at any Internet retailer's website.

Limitations and Directions for Future Research

Since collaborative filtering in the context of changing user preferences is a relatively new area of research, there are several open research directions.

We have presented a method to carry out collaborative filtering of transactional data or implicit rating data. This model can be extended to perform collaborative filtering of explicit rating data. We have done a limited examination of the effect of time period length on the performance of dynamic model. More study is needed to understand the relation between type of product and optimal length of the time period. We may find that a continuous time hidden Markov model avoids the problem of selecting the length of the time period at the cost of additional computation complexity.

Although not specific to dynamic models, using user and item attributes to improve recommender systems performance is an open research problem. Intuition suggests that using additional information in the form of such attributes should improve the recommendation quality. However, advantages of such additional data remains to be shown (Pilászy and Tikk 2009). In a separate study, we find that user and item attributes have a very interesting correlation with the class switching behavior of the users (Singh et al. 2010). However, deriving improved recommendations from this observation is another open research problem.

References

Adomavicius, G., and Tuzhilin, A. 2005. "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions," *IEEE Transactions on Knowledge and Data Engineering* (17:6), pp. 734-749.

Adomavicius, G., and Tuzhilin, A. 2011. "Context-Aware Recommender Systems," in *Recommender Systems Handbook,* F. Ricci, L. Rokach, B. Shapira and P. B. Kantor (eds.), New York: Springer, pp. 217-253.

Ansari, A., Essegaier, S., and Kohli, R. 2000. "Internet Recommendation Systems," *Journal of Marketing Research* (37:3), pp. 363-375.

Bell, R., and Koren, Y. 2007. "Lessons from the Netflix Prize Challenge," *ACM SIGKDD Explorations Newsletter* (9:2), pp. 75-79.

Bennett, J., and Lanning, S. 2007. "The Netflix Prize," KDD Cup and Workshop, Association for Computing Machinery, San Jose, California.

Billsus, D., and Pazzani, M. J. 1998. "Learning Collaborative Information Filters," in *Proceedings of the 15th International Conference on Machine Learning*, San Francisco: Morgan Kaufmann Publishers, pp. 46-54.

Bishop, C. 2006. *Pattern Recognition and Machine Learning*, New York: Springer.

Breese, J. S., Heckerman, D., and Kadie, C. 1998. "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, San Francisco: Morgan Kaufmann Publishers Inc., pp. 43-52.

Brusilovsky, P., Kobsa, A., and Nejdl, W. 2007. *The Adaptive Web: Methods and Strategies of Web Personalization*, Berlin: Springer-Verlag.

Celma, O. 2010. *Music Recommendation and Discovery: The Long Tail, Long Fail, and Long Play in the Digital Music Space*, New YorkL: Springer.

Chen, A. 2005. "Context-Aware Collaborative Filtering System: Predicting the User's Preference in the Ubiquitous Computing Environment," in *Location and Context-Awareness,* T. Strang and C. Linnhoff-Popien (eds.), Berlin: Springer, pp. 244-253.

Chien, Y. H., and George, E. I. 1999. "A Bayesian Model for Collaborative Filtering," in *Proceedings of the 7th International Workshop on Artificial Intelligence and Statistics*, San Francisco: Morgan Kaufmann Publishers, Inc.

Cunningham, P., Nowlan, N., Delany, S., and Haahr, M. 2003. "A Case-Based Approach to Spam Filtering that Can Track Concept Drift," in *Proceedings of the International Conference on Case-Based Reasoning*, New York: Springer, pp. 2003-2016.

Cutting, D., Kupiec, J., Pedersen, J., and Sibun, P. 1992. "A Practical Part-of-Speech Tagger," in *Proceedings of the Conference on Applied Natural Language Processing*, Association for Computational Linguistics, Trento, Italy, pp. 133-140.

Dahlquist, M., and Gray, S. F. 2000. "Regime-Switching and Interest Rates in the European Monetary System," *Journal of International Economics* (50:2), pp. 399-419.

Das, A. S., Datar, M., Garg, A., and Rajaram, S. 2007. "Google News Personalization: Scalable Online Collaborative Filtering," in *Proceedings of the 16th International Conference on World Wide Web*, Association for Computing Machinery, Banff, Alberta, Canada, pp. 271-280.

Dempster, A. P., Laird, N. M., and Rubin, D. B. 1977. "Maximum Likelihood from Incomplete Data Via the EM Algorithm," *Journal of the Royal Statistical Society* (39:1), pp. 1-38.

Deng, Y., and Byrne, W. 2008. "HMM Word and Phrase Alignment for Statistical Machine Translation," *IEEE Transactions on Audio, Speech, and Language Processing* (16:3), pp. 494-507.

Dunlavy, D. M., Kolda, T. G., and Acar, E. 2011. "Temporal Link Prediction Using Matrix and Tensor Factorizations," *ACM Transactions on Knowledge Discovery from Data* (5:2), pp. 1-27.

Fleder, D., and Hosanagar, K. 2009. "Blockbuster Culture's Next Rise or Fall: The Impact of Recommender Systems on Sales Diversity," *Management Science* (55:5), pp. 697-712.

Getoor, L., and Sahami, M. 1999. "Using Probabilistic Relational Models for Collaborative Filtering," paper presented to the Workshop on Web Usage Analysis and User Profiling, San Diego, CA.

Goldberg, D., Nichols, D., Oki, B., and Terry, D. 1992. "Using Collaborative Filtering to Weave an Information Tapestry," *Communications of the ACM* (35:12), pp. 61-70.

Hamilton, J. D. 1989. "A New Approach to the Economic Analysis of Nonstationary Time Series and the Business Cycle," *Econometrica: Journal of the Econometric Society* (57:2), pp. 357-384.

Harries, M., Sammut, C., and Horn, K. 1998. "Extracting Hidden Context," *Machine learning* (32:2), pp. 101-126.

Heckerman, D., Chickering, D. M., Meek, C., Rounthwaite, R., and Kadie, C. 2001. "Dependency Networks for Inference, Collaborative Filtering, and Data Visualization," *The Journal of Machine Learning Research* (1), pp. 49-75.

Herlocker, J., Konstan, J., Borchers, A., and Riedl, J. 1999. "An Algorithmic Framework for Performing Collaborative Filtering," in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval,* Association for Computing Machinery, Berkeley, CA, pp. 230-237.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. 2004. "Evaluating Collaborative Filtering Recommender Systems," *ACM Transactions on Information Systems* (22:1), pp. 5-53.

Hofmann, T. 2004. "Latent Semantic Models for Collaborative Filtering," *ACM Transactions on Information Systems* (22:1), pp. 89-115.

Hofmann, T., and Puzicha, J. 1999. "Latent Class Models for Collaborative Filtering," in *Proceedings of the 16tth International Joint Conference on Artificial Intelligence*, San Francisco: Morgan Kaufmann Publishers, pp. 688-693.

Hu, Y., Koren, Y., and Volinsky, C. 2008. "Collaborative Filtering for Implicit Feedback Datasets," in *Proceedings of the IEEE International Conference on Data Mining*, Pisa, Italy, pp. 263-272.

Huang, Z., Zeng, D. D., and Chen, H. 2007a. "Analyzing Consumer-Product Graphs: Empirical Findings and Applications in Recommender Systems," *Management science* (53:7), pp. 1146-1164.

Huang, Z., Zeng, D., and Chen, H. 2007b. "A Comparison of Collaborative-Filtering Recommendation Algorithms for E-commerce," *IEEE Intelligent Systems* (22:5), pp. 68-78.

Järvelin, K., and Kekäläinen, J. 2002. "Cumulated Gain-Based Evaluation of IR Techniques," *ACM Transactions on Information Systems* (20:4), pp. 422-446.

Juang, B., and Rabiner, L. 1991. "Hidden Markov Models for Speech Recognition," *Technometrics* (33:3), pp. 251-272.

Karlof, C., and Wagner, D. 2003. "Hidden Markov Model Cryptanalysis," in *Cryptographic Hardware and Embedded Systems,* C. Walter, Ç. Koç and C. Paar (eds.), Berlin: Springer, pp. 17-34.

Koren, Y. 2009. "The Bellkor Solution to the Netflix Grand Prize," Netflix prize documentation.

Koren, Y. 2010. "Collaborative Filtering with Temporal Dynamics," *Communications of the ACM* (53:4), pp. 89-97.

Koren, Y., Bell, R., and Volinsky, C. 2009. "Matrix Factorization Techniques for Recommender Systems," *IEEE Computer* (42:8), pp. 30-37.

Lang, K. 1995. "Newsweeder: Learning to Filter Netnews," in *Proceedings of the International Conference on Machine Learning*, San Francisco: Morgan Kaufmann, pp. 331--339.

Levinson, S. 1986. "Continuously Variable Duration Hidden Markov Models for Automatic Speech Recognition," *Computer Speech & Language* (1:1), pp. 29-45.

Liben Nowell, D., and Kleinberg, J. 2007. "The Link Prediction Problem for Social Networks," *Journal of the American Society for Information Science and Technology* (58:7), pp. 1019-1031.

Lu, H. M., Zeng, D., and Chen, H. 2010. "Prospective Infectious Disease Outbreak Detection Using Markov Switching Models," *IEEE Transactions on Knowledge and Data Engineering* (22:4), pp. 565-577.

Lukashin, A., and Borodovsky, M. 1998. "GeneMark.hmm: New Solutions for Gene Finding," *Nucleic Acids Research* (26:4), pp. 1107-1115.

Macskassy, S., and Provost, F. 2004. "Confidence Bands for ROC Curves: Methods and an Empirical Study," in *Proceedings of the First Workshop on ROC Analysis in AI*, Valencia, Spain, pp. 61-70.

Minka, T. P. 2002. "Estimating a Gamma Distribution," unpublished paper (http://research.microsoft.com/en-us/um/people/minka/papers/minka-gamma.pdf).

Mooney, R. J., and Roy, L. 2000. "Content-Based Book Recommending Using Learning for Text Categorization," in *Proceedings of the 5th ACM Conference on Digital Libraries*, San Antonio, TX, pp. 195-204.

Murphy, K. P. 2002. *Dynamic Bayesian Networks: Representation, Inference and Learning*, unpublished Ph.D. thesis, University of California, Berkley (www.ai.mit.edu/~murphyk/Thesis/thesis.pdf).

Netzer, O., Lattin, J., and Srinivasan, V. 2008. "A Hidden Markov Model of Customer Relationship Dynamics," *Marketing Science* (27:2), pp. 185-204.

Notredame, C. 2002. "Recent Progress in Multiple Sequence Alignment: A Survey," *Pharmacogenomics* (3:1), pp. 131-144.

Ostendorf, M., Digalakis, V. V., and Kimball, O. A. 1996. "From HMM's to Segment Models: A Unified View of Stochastic Modeling for Speech Recognition," *IEEE Transactions on Speech and Audio Processing* (4:5), pp. 360-378.

Pan, R., and Scholz, M. 2009. "Mind the Gaps: Weighting the Unknown in Large-Scale One-Class Collaborative Filtering," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Paris, France, pp. 667-676.

Paterek, A. 2007. "Improving Regularized Singular Value Decomposition for Collaborative Filtering," *KDDCup.07*, Association for Computing Machinery, San Jose, CA, August 12, pp. 39-42 (http://www.cs.uic.edu/~liub/KDD-cup-2007/proceedings/Regular-Paterek.pdf).

Pazzani, M., Muramatsu, J., and Billsus, D. 1996. "Syskill & Webert: Identifying Interesting Web Sites," paper presented at the AAAI Spring Symposium, Stanford, CA (http://www.cs.rutgers.edu/~pazzani/Publications/Syskill.pdf).

Pilászy, I., and Tikk, D. 2009. "Recommending New Movies: Even a Few Ratings are More Valuable than Metadata," in *Proceedings of the 3rd ACM Conference on Recommender Systems*, Association for Computing Machinery, pp. 93-100.

Rabiner, L. 1989. "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE* (77:2), pp. 257-286.

Rashid, A., Albert, I., Cosley, D., Lam, S., McNee, S., Konstan, J., and Riedl, J. 2002. "Getting to Know You: Learning New User Preferences in Recommender Systems," in *Proceedings of the International Conference on Intelligent User Interfaces*, Association for Computing Machinery, pp. 127-134.

Rendle, S., Freudenthaler, C., and Schmidt-Thieme, L. 2010. "Factorizing Personalized Markov Chains for Next-Basket Recommendation," in *Proceedings of the 19th International Conference on World Wide Web*, Association for Computing Machinery, pp. 811-820.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. 1994. "GroupLens: An Open Architecture for Collaborative Filtering of Netnews," in *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, Association for Computing Machinery, pp. 175-186.

Resnick, P., and Varian, H. R. 1997. "Recommender Systems," *Communications of the ACM* (40:3), pp. 56-58.

Sahoo, N., Krishnan, R., Duncan, G., and Callan, J. 2012. "The Halo Effect in Multicomponent Ratings and its Implications for Recommender Systems: The Case of Yahoo! Movies," *Information Systems Research* (23:1), pp. 231-246.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. 2001. "Item-Based Collaborative Filtering Recommendation Algorithms," in *Proceedings of the 10th international conference on World Wide Web*, Association for Computing Machinery, pp. 285-295.

Schlimmer, J., and Granger, R. 1986. "Beyond Incremental Processing: Tracking Concept Drift," in *Proceedings of the 5th National Conference on Artificial Intelligence*, Philadelphia, PA, pp. 502-507.

Shardanand, U., and Maes, P. 1995. "Social Information Filtering: Algorithms for Automating 'Word of Mouth,'" in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Association for Computing Machinery, pp. 210-217.

Si, L., and Jin, R. 2003. "Flexible Mixture Model for Collaborative Filtering," in *Proceedings of the International Conference on Machine Learning*, Washington, DC: AAAI Press, pp. 704-711.

Singh, P. V., Sahoo, N., and Mukhopadhyay, T. 2010. "Seeking Variety: A Dynamic Model of Employee Blog Reading Behavior," unpublished paper, Carnegie Mellon University.

Singh, P. V., Tan, Y., and Youn, N. 2011. "A Hidden Markov Model of Developer Learning Dynamics in Open Source Software Projects," *Information Systems Research* (22:4) pp. 790-807.

Street, W. N., and Kim, Y. 2001. "A Streaming Ensemble Algorithm (SEA) for Large-Scale Classification," in *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Association for Computing Machinery, pp. 377-382.

Swets, J. A. 1963. "Information Retrieval Systems," *Science* (141:3577), pp. 245-250.

Tsymbal, A. 2004. "The Problem of Concept Drift: Definitions and Related Work," unpublished paper, Computer Science Department, Trinity College, Dublin.

Ungar, L. H., and Foster, D. P. 1998. "Clustering Methods for Collaborative Filtering," in *Proceedings of the AAAI Workshop on Recommendation Systems*, pp. 112-125.

van Setten, M., Pokraev, S., and Koolwaaij, J. 2004. "Context-Aware Recommendations in the Mobile Tourist Application COMPASS," in *Adaptive Hypermedia and Adaptive Web-Based Systems,* P. De Bra and W. Nejdl (eds.), Berlin: Springer, pp. 515-548.

Widmer, G., and Kubat, M. 1996. "Learning in the Presence of Concept Drift and Hidden Contexts," *Machine Learning* (23:1), pp. 69-101.

Xiang, L., Yuan, Q., Zhao, S., Chen, L., Zhang, X., Yang, Q., and Sun, J. 2010. "Temporal Recommendation on Graphs via Long- and Short-term Preference Fusion," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Association for Computing Machinery, pp. 723-732.

About the Authors

Nachiketa Sahoo is an assistant professor of Information Systems at the School of Management, Boston University. His research interest lies at the intersection of statistical machine learning and social sciences. He is currently doing research on personalized recommender systems and expertise discovery strategies on social media. His research has been published in *Information Systems Research* as well as in the proceedings of conferences in computer science and information systems.

Param Vir Singh is an assistant professor of Information Systems at the David A Tepper School of Business, Carnegie Mellon University. Param's research interests entail understanding the underlying micro foundations of the online communities (both within and outside firms) formed around Web 2.0/social media technologies. His research goals are to provide policy and design implications for these communities to help them achieve the goals for which these media were created. To address design and policy questions, he builds dynamic structural models of individual behavior and conducts counterfactuals and policy simulations to analyze the impacts of interventions on measures of economic interest. His papers have been published in various journals in Information Systems area including *MIS Quarterly* and *Information Systems Research*.

Tridas Mukhopadhyay is Deloitte Consulting Professor of e-Business at Carnegie Mellon University. He received his Ph.D. in Computer and Information Systems from the University of Michigan in 1987. His research interests include strategic use of IT, electronic commerce, business value of information technology, and software development productivity.

MIS Quarterly

SPECIAL ISSUE: BUSINESS INTELLIGENCE RESEARCH

A HIDDEN MARKOV MODEL FOR COLLABORATIVE FILTERING

Nachiketa Sahoo
School of Management, Boston University, 595 Commonwealth Avenue, Boston, MA 02215 U.S.A.
and iLab, Heinz College, Carnegie Mellon University, Pittsburgh, PA 15213 U.S.A. {nachi@bu.edu}

Param Vir Singh
David A. Tepper School of Business and iLab, Heinz College, Carnegie Mellon University,
Pittsburgh, PA 15213 U.S.A. {psidhu@cmu.edu}

Tridas Mukhopadhyay
David A. Tepper School of Business and iLab, Heinz College, Carnegie Mellon University,
Pittsburgh, PA 15213 U.S.A. {tridas@cmu.edu}

Appendix A

Prediction Distribution over Items

The probability of a user u selecting an item i in time period t is

$$P(i \in I_u^t) = \sum_k P(Z_u^t = k) P(i \in I_u^t; a_k, b_k, \theta_k) \tag{1}$$

where, I_u^t is the set of items selected by the user u in time period t.

Conditional on the user being in state the probability of the item being observed is

$$P(i \in I_u^t; a_k, b_k, \theta_k) = \sum_{N_u^t=0}^{\infty} P(N_u^t; a_k, b_k) \times P(i \in I_u^t | N_u^t; \theta_k) \tag{2}$$

This follows from considering the probability of each possible number of items (N_u^t) selected by the user and for each of those number of items selected the probability of the item being included in the selected set.

Since the distribution over the items is a multinomial, the probability that an item i is observed in the N_u^t items that are observed in time period t is

$$P(i \in I_u^t | N_u^t; \theta_k) = 1 - P(i \notin I_u^t | N_u^t; \theta_k) = 1 - (1 - P(i|\theta_k)^{N_u^t} = 1 - (1 - \theta_{ki})^{N_u^t} \tag{3}$$

The last equality of equation (3) follows from the fact that the probability of any item i for in a multinomial is equal to the parameter of the multinomial specific to that item. Substituting equation (3) in Equation (2) we get

$$\begin{aligned} P(i \in I_u^t; a_k, b_k, \theta_k) &= \sum_{N_u^{t+1}=0}^{\infty} P(N_u^t; a_k, b_k) \times \left(1 - (1-\theta_{ki})^{N_u^t}\right) \\ &= 1 - \sum_{N_u^{t+1}=0}^{\infty} P(N_u^t; a_k, b_k) \times (1-\theta_{ki})^{N_u^t} \end{aligned} \tag{4}$$

This summation, which is expectation of the exponential function of $N_u^t \ln(1 - \theta_{ki})$, can be obtained using the identity for moment generating function for the negative binomial distribution.

$$\left(e^{Xt}\right)_{x \sim P_{NDBD(a,b)}} = \left\{1 + b(1 - e^t)^{-a}\right\} \tag{5}$$

Substituting $\ln(1 - \theta_{ki})$ for t in equation (5) and using the results in equation (4) after some algebraic reduction, we obtain

$$P(i \in I_u^t; a_k, b_k, \theta_k) = 1 - (1 + b_k \theta_{ki})^{-a_k} \tag{6}$$

Substituting equation (6) in equation (1), we obtain the following expression for the probability of observing the item i

$$P(i \in I_u^t) = \sum_k P(Z_u^t = k)\left(1 - (1 + b_k \theta_{ki})^{-a_k}\right) = 1 - \sum_k P(Z_u^t = k)(1 + b_k \theta_{ki})^{-a_k} \tag{7}$$

For ordering items by their probability of occurrence we can drop the unity and order the items by $-\sum_k P(Z_u^t = k)(1 + b_k \theta_{ki})^{-a_k}$.

The distribution over the states for a user can be calculated from the user's distribution in the previous time period as $P(Z_u^t = k) = \sum_{l=1}^{K} P(Z_u^{t-1} = l) P(Z_u^t = k | Z_u^{t-1} = l)$, where K is the number of possible states.

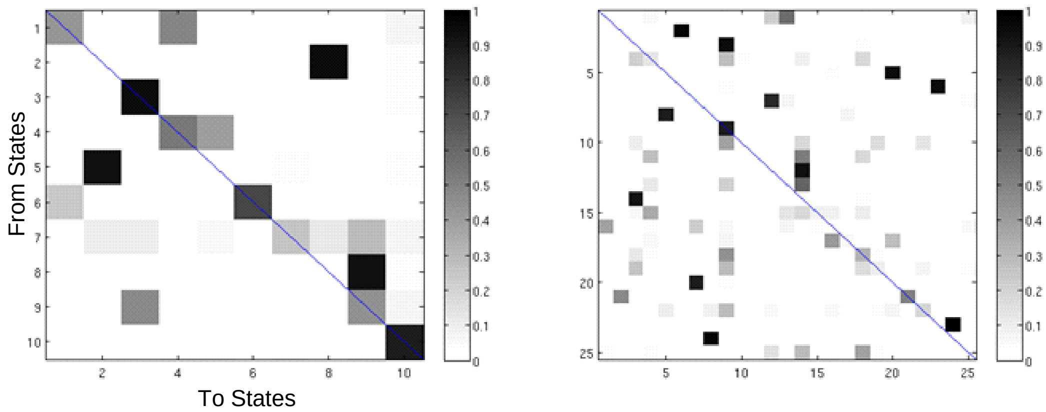
Appendix B

Latent Classes and Transitions in Blog Data

Upon examination of the transition probability matrix we find that, although there are classes from which the users do not move, there are many classes from which the users tend to switch to other classes in the subsequent time period. This behavior is illustrated in Figure B1.

The latent classes can be distinguished by their different intensity of reading and by the items that are the most popular in the latent class. These are reported in Table B1 for the HMM with 10 latent classes. As we can see, they differ in how much a user reads when the user is under a given latent class. Although there is some overlap in the top articles read by the users in each class, they have several differences in their selection of favorite articles to read.

Note that the class under which the users read the most, class 7, is also the one from which they switch away to a less active class, such as class 8. This suggests that it is improbable that the users will stay in a highly active class for long. On the other hand, the users tend to stay longer in a class that is less active, for example, class 10.



Notes: The transition probability matrix shown as gray scale images for HMM with 10 latent classes (left) and 25 latent classes (right). The darker the cell color is, the larger the transition probability. Although there are a few large probabilities on the diagonal, there are several large off diagonal probabilities suggesting a class switching behavior for the user.

Figure B1. Transition Probability Matrix

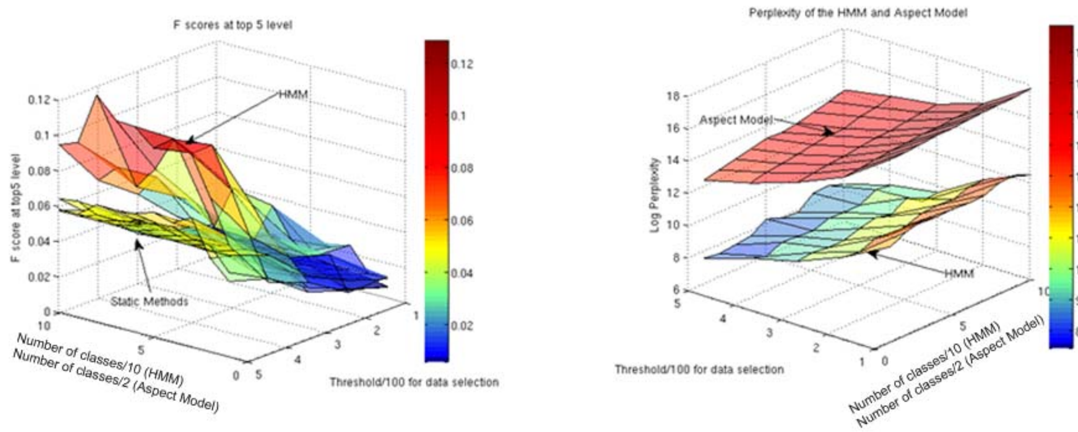
Table B1. Latent Classes[†]

Latent Class	Average number of articles read in a month	Index of the top 5 most probable articles to be read				
		#1	#2	#3	#4	#5
1	3.9121	233	107	113	223	126
2	6.6621	21	39	52	230	46
3	1.3722	233	223	107	167	80
4	3.4601	233	39	253	42	178
5	6.8933	39	42	150	36	230
6	1.7879	126	223	233	113	107
7	17.5808	39	31	156	3	187
8	5.4393	39	102	52	156	126
9	2.6996	39	52	107	126	205
10	0.0396	126	233	108	113	104

[†]The latent classes are characterized here by the average number of posts read by a user in the class in a month and the top articles read in the class.

Appendix C

Sensitivity Analysis with Blog Data



Notes: Sensitivity analysis is performed by varying the data densities and the number of classes and factors. The number of classes used for HMM was varied from 10 to 100 and from 2 to 20 for the static models. The data density was controlled by varying the minimum number of articles a user must visit and the minimum number of users an article must be read by from 100 to 500. The “Number of Classes” axis shows (number of classes/10) used for the HMM and (number of classes/2) used for static models. The “Threshold” axis shows the (minimum threshold/100) used for all the methods. The *F-score* when the top 5 articles are recommended is shown in the left plot. The *perplexity* of the HMM and aspect model are compared in the right plot.

Figure C1. Sensitivity Analysis