# A Hidden Markov Model for Collaborative Filtering


| Nachiketa Sahoo | Param Vir Singh | Tridas Mukhopadhyay |
|---|---|---|
| Tepper School of Business and iLab, Heinz College Carnegie Mellon University | Tepper School of Business and iLab, Heinz College Carnegie Mellon University | Tepper School of Business and iLab, Heinz College Carnegie Mellon University |



## Abstract

We present a hidden Markov model for collaborative filtering of implicit ratings when the ratings have been generated by a set of changing user preferences. Most of the works in the collaborative filtering and recommender systems literature have been developed under the assumption that user preference is a static pattern. However, we show by analyzing a dataset on employees' blog reading behaviors that users' reading behaviors do change over time. We model the unobserved user preference as a Hidden Markov sequence. The observation that users read variable numbers of blog articles in each time period and choose different types of articles to read, requires a novel observation model. We use a Negative Binomial mixture of Multinomials to model such observations. This allows us to identify stable global preferences of users towards the items in the dataset and allows us to track the users through these preferences. We compare the algorithm with a number of static algorithms and a recently proposed dynamic collaborative filtering algorithm and find that the proposed HMM based collaborative filter outperforms the other algorithms.

# A Hidden Markov Model for Collaborative Filtering

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

## Introduction

### 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 the 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. 2010), 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 be static patterns (Adomavicius and Tuzhilin 2005). However, common experience suggests that user preferences 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, 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 the customers' preferences from their past ratings on products. Such a system might have successfully identified the consumers' prior preferences; 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 algorithm during the training phase. By fitting a static model to data generated by a dynamic process one learns a mis-specified model. Therefore, 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 that allows one to account for changing user preferences which is not possible in the traditional recommender system models. We examine the challenges related to interpreting time-stamped ratings and learning from them when they are contributed by users whose preferences evolve over time. To overcome these challenges we present a Hidden Markov model with a novel emission component. The model learns the global preference patterns that can be used to make personalized recommendations in a time sensitive manner. 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 the other users of the system have rated. The inputs to the system are the records of data containing user id, item id, and the ratings the user has provided for the item. By providing ratings on items, the 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 sharp 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 simple data requirements, i.e., they do not need data on the properties of the items or demographic characteristics of the users. In addition, unlike the content based approaches, they are not limited to recommending only those items with attributes matching the items a user has liked in the past. Therefore, they have been a popular choice for use in recommender systems.

The first group of collaborative filtering algorithms were primarily instance based (Resnick et al. 1994b). In the training step they 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 datasets. However, they do not scale to large datasets 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 the 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 $1M for developing an algorithm that predicts Netflix-users' ratings on movies by at least 10% more accurately than the existing Cinematch movie recommender 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 a set of columns of user weights—one column for each factor. Similarly the Item factor matrix consists of a set of columns of item weights. The weights indicate the degree of membership of the user and the item into different latent groups.

Most of the collaborative filtering algorithms are based upon the assumption that user preference is a static pattern. The task of the algorithms 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. This is a rather strong assumption especially with the consumption of certain classes of products that occur over a long time period. User preferences often evolve with the age of the user 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 rating on items leads to better estimation of the item ratings (Koren 2010). This algorithm, known as *timeSVD++*, identifies transient patterns that existed among the ratings collected in the past so that persistent patterns about the user preferences can be isolated and used

for recommendation. 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, they use tensor factorization to isolate a few top factors that describe dominant transition behavior. Despite these recent interests in modeling user feedback as a dynamic event this remains a relatively less explored topic in collaborative filtering literature.

## Context-aware recommendation

Often how the user rates 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 specific 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, recommending movies to watch depending on the day of the week, etc. There are two related strategies that such systems use 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 still estimate the user preference as a static function, i.e., unchanging with time. However in the current paper we want to model the internal evolution of a user's preference over time.

## Explicit vs. 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 by how much, by providing a rating on a scale of e.g. 1—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 clickstreams 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 of such implicit ratings (Hu et al. 2009). With transactional data we observe a user selecting an item but we do not know if the user liked or, more importantly, 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 dataset consists of such bookmarks we do not have any negative data points to learn from. 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 dataset 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, Recall etc. have been used to compare algorithms that use implicit ratings (Huang et al. 2007).

## 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 next. Market conditions and moods often have yearly or even monthly recurring patterns. The nature of spam emails 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 so that reliable predictions can be made for 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 dataset 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). Similar approaches have 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 a

cluster of quality metric to decide 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 learnt 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 evidence from discarding old data often outweighs any benefit from the removal of irrelevant data. As a result, ignoring old data results in inferior recommendations (Koren 2010).

## Dynamic models

There is a rich stream of literature on statistical dynamic models. The two most closely related streams involve hidden Markov models and regime switching models. A hidden Markov model (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). It has 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*, models 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 (Kevin 2002).

The regime switching models differ from the hidden Markov models in that the regime switching is considered to be exogenous and observed whereas in hidden Markov models it is assumed to follow a Markov process. The regime switching models typically model the change in behavior of an actor in response to an event. The regime switching models have found application in dynamics of economic actors in response to events such as financial crisis (Cerra and Saxena 2005; Hamilton 2005; Jeanne and Masson 2000) or abrupt changes in government policy (Hamilton 1988; Sims and Zha 2006).

Despite this rich body of literature in dynamic models there has been little research in 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

## Context, dataset and the task

The context of this research is a corporate blog network. The increasing adoption of Web 2.0 technologies and cultures within enterprises is encouraging employees in the firm 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 the firm because it gives the users access to the expertise of other employees. However, it also creates an information overload. Although there could be thousands of articles written by the employees, finding the relevant article for a particular employee is not a trivial task. In this work we focus on the corporate blog network of a large Fortune 500 IT services firm and propose a recommender system to alleviate the information overload.

To train the recommender system we collect the log of the users' visits to blog articles from the webserver that hosts the blogs. This access log provides us implicit ratings of the users on the blog articles. The dataset was collected over 22 months (Jan '07—Oct '08). There were 71.5k articles posted during this period. The articles were read by 52k employees. There are 2.2 million time stamped visits by the blog readers to the blog articles. The articles have been classified into 25 predefined topics by their authors. Some of the topics are "Knowledge Management", "Software Testing", "Photography", etc.

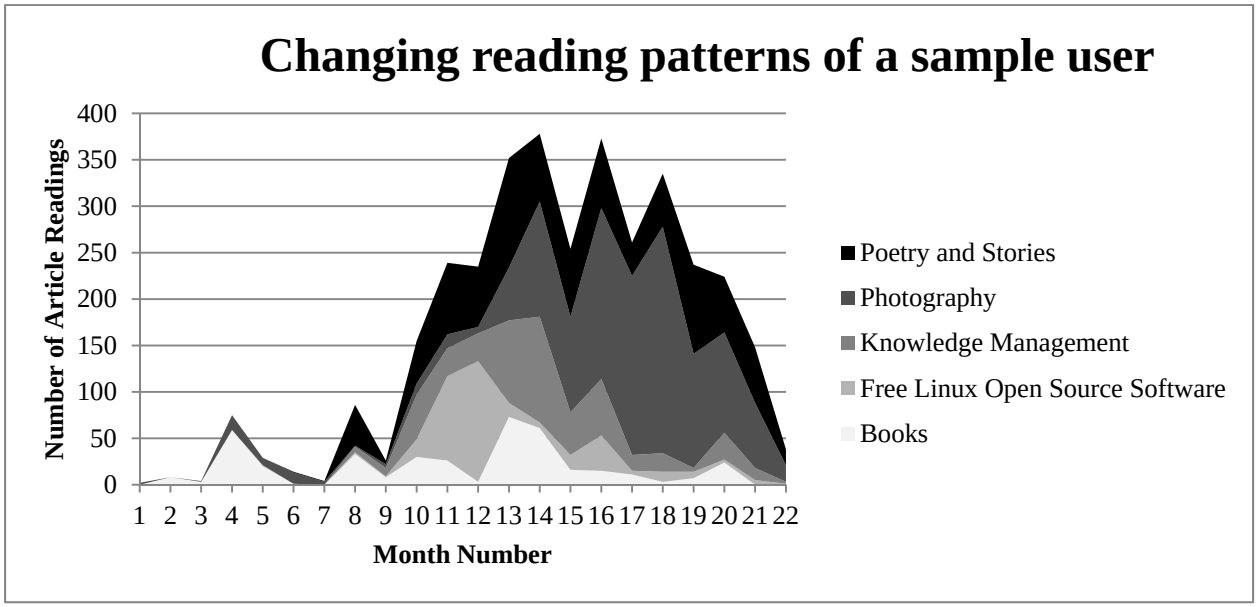


**Figure 1. A stacked plot of the number of articles read by a sample user in five different topics over the months.**

The static recommender systems produce recommendations that the user might like without any reference to the time period when the user might like them. However, an examination of the blog reading behavior

shows that blog readers change the amount and the type of posts they read over time. In Figure 1 we show the volume of articles read by a random user in five different topics over the 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 the 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—11, the user starts reading in the topic of "Linux and Open Source Software" and also in "Knowledge Management" and "Poetry & Stories." Although subsequently the user reduces reading in the topic of "Linux and Open Source Software," the reader continues to read articles in "Poetries and Stories" and slowly reduces reading "Knowledge Management" related posts. Around months 12—13 the user starts reading Photography related posts, which continues to dominate most of his reading activity in the subsequent months.

This observation suggests that users' reading preference, as evidenced by the type of posts they read, is not a static pattern, but a continuously changing pattern. Therefore, in order to provide a solution to the information overload, the task of practical importance is to make personalized article 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 the 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 for training and the task of the algorithm is to predict the future ratings, when the user's preferences might have changed.

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 not at any time in the future, but, in the next time period. It means that during evaluation the algorithm must successfully identify a smaller set of items that the user selects in a given time 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 want to use. We evaluate the algorithms for their performance in time specific recommendation. In our experiments the length of each time period was set to one month.

## Issues to be addressed

Let's consider the instance based static collaborative filtering algorithm based on the similarity between pairs of users (Shardanand and Maes 1995). Each user's fixed preference is represented by the ratings she has provided on a set of items. Then a multi-dimensional distance metric is used to compute the similarity between pairs of users. This approach breaks down when the preference changes over time. The rating data for each user is not generated from one fixed unknown preference, but, from a series of unknown preferences. Thus it is not clear how one can measure the similarity between two users. Even if it were possible, 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 preferences, and preferences ultimately determine whether a user is going to like the recommended item or not.

A parallel issue exists for the 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 these models 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 the users and items because of the assumption that all the selections of the items by the 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. A model based algorithm might incorporate such change as a changing degree of membership to different classes for each user. But, how the observation from each time period can be used to determine these user classes has not been explored in the literature.

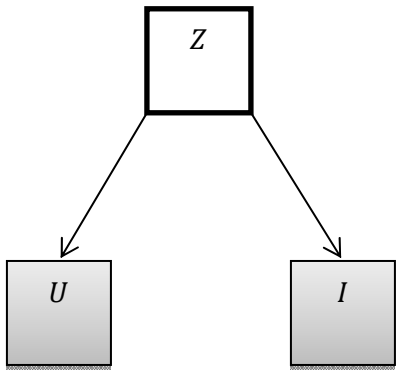


**Figure 2 Aspect Model of User-Item co-occurrence. The user (U) and item (I) occurrences are governed by a latent class (Z). Each data row is of the form (User ID, Item ID). In this and the remaining Bayesian Network representations of the models in this paper the observed variables are represented by a filled-in square and latent variables are represented by an empty square.**

Our task would be considerably simpler if we could get repeated ratings from the user on a product. This would allow us to infer how the user's preference has changed. For example if a person rated the movie *Die Hard* 5, on a scale of 1—5, when he was 20 years old, but rates the movie 3 when he is 21, we would conclude that the person does not like action/adventure movies as much[1]. However, such data is rarely available in practice. Often each rating is on a unique item. Therefore, our only means of inferring any shift in the user's preference is from the user's ratings on unique items. In datasets that do not contain explicit ratings, but only the selection of items by the users, the possibility of multiple selections is ruled out. We automatically have a single observation for each user-item pair. The task in this case becomes not to predict the rating a user will give an item, but, to estimate the probability of the user selecting the item. This leads to the other challenge of modeling the users' item selection behavior.

Not only the types of the items users select changes, but the number of items they select in each time period may also change. The number of items they select in a given time period affects the probability of their selecting any particular item in that period. Therefore, to estimate the probability of a user selecting a particular item in a future time period in our chosen blog context, we need to be able to predict the number of posts he is going to read as well.

These issues lead us to a set of research questions that need to be addressed to build personalized recommender systems while user behavior changes.

### Research questions

1. How can the old user ratings, generated by a prior temporary user preference, be used to estimate global 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 he is reading, but how much he is reading as well?

## A Hidden Markov Model of User Preference

We design a model of changing user preference based on the probabilistic graphical modeling framework. Therefore, it is helpful to start by examining a static model based on this framework. Let's consider the Aspect Model (Figure 2). In this model the distribution over users and items is modeled as $P(U, I) =$

[1] Of course, it would not be wise to conclude a preference shift from one such observation. However, a series of such observations on related movies such as *Star Wars*, *Indiana Jones*, etc. would help infer the shift in the person's movie preference.

$\sum_Z P(Z)P(U|Z)P(I|Z) = \sum_Z P(U)P(Z|U)P(I|Z)$, i.e., the occurrence of Item in a data row is independent of the occurrence of the user if we know the distribution over the occurrence of the latent class. So, if we are interested in predicting the occurrence of an item in a data record, 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 $I$. The entire preference of a user for different items is encoded in the user's membership in 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 ($\boldsymbol{\pi}$).
2. The *transition probability* table between classes in adjacent time periods ($\mathbf{A}$).
3. The *emission* or *observation* model that generates the data from the latent class memberships in each time period.

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. 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 up until time period $t-1$ are taken into account to compute the user's membership in the latent classes in time period $t$, which is used to predict the items 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 in 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 in different latent classes allows us to model each user's visits to blog articles in a unique way.

The starting probability distribution and the transition probability distribution have a lot of similarity with the ones proposed 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 distribution* as a set of class specific Negative Binomial distributions, each of which has a pair of parameters $(a_k, b_k)$.
2. A distribution determining what articles will be visited in that month: $\{I_{uj}^t\}$. We model this *item selection* distribution as a set of class specific Multinomial distribution each with a vector parameter $\boldsymbol{\theta_k}$, which is a probability distribution over all the items. Each of the $N_u^t$ articles is assumed to be drawn from the class specific Multinomial distribution.

We use the formulation of Negative Binomial Distribution as a Gamma Mixture of Poisson:

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

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

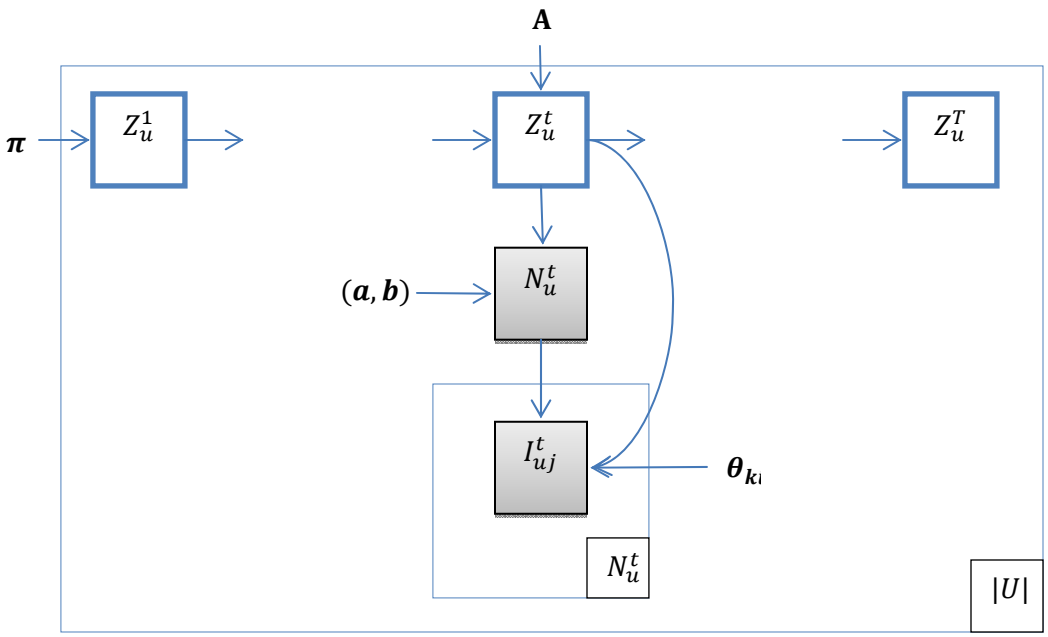


**Figure 3 An HMM for the blog reading behavior of the users.**

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 $I_{uj}^t$ are the observed variables. $N_u^t$ is the number of articles the user reads in time period $t$. $I_{uj}^t$ is the ID of the $j$th article the user $u$ reads in time period $t$. $|U|$ is the number of users in the dataset and $|I|$ is the number of items in the dataset. The parameters of the model are $\boldsymbol{\pi}, \mathbf{A}, \boldsymbol{a}, \boldsymbol{b},$ and $\boldsymbol{\theta_k}$. If there are $K$ latent classes then these parameters are vectors and matrices of sizes $k \times 1, k \times k, k \times 1, k \times 1,$ and $|I| \times 1$ respectively.

## Estimation and Complexity

The parameters were estimated by the Expectation Maximization (EM) approach (Dempster et al. 1977). In this approach the parameters are initialized to some values and then optimized via two alternating steps:

1. **Expectation step**: The distribution over the hidden variable, $Z_u^t$, is computed for each user using the values of the parameters obtained so far and the observations for the user.
2. **Maximization 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).

The **expectation** step amounts to performing *inference* on the latent variables given the observations using the current estimates of the parameters. Let's assume that $X$ is the set of observed variables, $Z$ is the set of latent variables, and $\Theta$ is the set of all parameters. Inferring the latent variables would amount to computing the conditional distribution over the latent variables, which can be written as:

$$P(Z|X;\Theta) = \frac{P(X,Z;\Theta)}{P(X;\Theta)} = \frac{P(X,Z;\Theta)}{\sum_Z P(X,Z;\Theta)}$$

Most of the complexity of this calculation comes from the summation to marginalize $Z$. For distributions that can be represented by a graphical model with a tree structure, this summation can be done in linear time using a sum-product algorithm (Kschischang et al. 2001). The number of parameters that needs to be estimated for the transition matrix grows as the square of the number of classes $K$. Since HMMs are of chain structure the algorithm is also known as the *Forward-Backward* algorithm (Rabiner 1989). With such a chain structure Expectation step can be done in $O(TK^2)$ time using the *Forward-Backward* algorithm.

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

$$\sum_Z P\left(Z\middle|X;\Theta^{old}\right)\log P(X,Z;\Theta) \leq \log P(X;\Theta) \tag{2}$$

This describes the lower bound for one observation. If there are multiple instances drawn from the same model then we have to compute the value of $\Theta$ that maximizes the sum of expected log probabilities: $\sum_i \sum_Z P\left(Z_i\middle|X_i;\Theta^{old}\right)\log P(X_i,Z_i;\Theta)$.

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

$$\begin{aligned}&\sum_k P\left(Z_u^1 = k\middle|X;\Theta^{old}\right)\log \pi_k \\ &+\sum_{t=2}^{T}\sum_j\sum_k P(Z_u^{t-1}=j, Z_u^t = k\middle|X;\Theta^{old})\log A_{jk} \\ &+\sum_{t=1}^{T}\sum_k P\left(Z_u^t = k\middle|X;\Theta^{old}\right)\log P(N_u^t,\{I_{uj}^t\}|\, a_k, b_k, \boldsymbol{\theta_k})\end{aligned} \tag{3}$$

Note that parameters for the three distributions can be maximized independent of each other. The calculation for the optimal $\pi_k$ and $A_{jk}$ can be borrowed from the literature (Bishop 2006). Maximizing Expression (3) leads to the Maximum Likelihood Estimates of the parameters. However, MLEs run the

risk of over fitting when the size of the training dataset 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 and maximized. 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 parameter value that leads to maximum posterior probability.

The distribution of a user's latent class at $t = 1$ and the conditional distribution of the user's latent class at $t > 1$ are multinomial. Therefore, the conjugate prior distribution from which the parameters $\pi_k$ and $A_{jk}$ are drawn are Dirichlet distributions. This leads to closed form MAP estimates of the parameters. We use the following Dirichlet prior for all the Multinomial distributions.

$$\boldsymbol{\pi}, \boldsymbol{A_{j,:}} \sim Dir(\boldsymbol{x}|\alpha_1, \dots, \alpha_K) \tag{4}$$

where, each $\alpha_k$is set to $\frac{\alpha}{K}$. The weight of the evidence provided by each prior is $\alpha$. The MAPs of the parameters are given by:

$$\widehat{\pi_k} = \frac{\left[\sum_u P\left(Z_u^1 = k \middle| X; \Theta^{old}\right)\right] + \alpha_k - 1}{\left[\sum_u \sum_k P(Z_u^1 = k|X; \Theta^{old})\right] + \alpha - K} \tag{5}$$

$$\widehat{A_{jk}} = \frac{\sum_t^T P\left(Z_u^{t-1} = j, Z_u^t = k \middle| X; \Theta^{old}\right) + \alpha_k - 1}{\sum_t^T \sum_l P(Z_u^{t-1} = j, Z_u^t = l|X; \Theta^{old}) + \alpha - K} \tag{6}$$

The emission model is unique. Each class specific emission distribution is a Negative Binomial mixture of Multinomial distributions. From Figure 3 the observation $N_u^t$ d-separates $(a_k, b_k)$ from $\boldsymbol{\theta_k}$**.** Therefore, log likelihood of the parameters for the emission distribution decomposes as sum of functions of these two sets of parameters, and so does their expectation. Expected log likelihood of the observation model is:

$$\begin{aligned} &\sum_t^T \sum_k P\left(Z_u^t = k \middle| X; \Theta^{old}\right) \log P(N_u^t | \, a_k, b_k) \\ &+ \sum_t^T \sum_k P\left(Z_u^t = k \middle| X; \Theta^{old}\right) \log P\left(\left\{I_{uj}^t\right\} \middle| \boldsymbol{\theta_k}, N_u^t\right) \end{aligned} \tag{7}$$

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, which is similar to Equations (5) and (6). However, there is no closed form solution for calculating the MAP estimate of the weighted Negative Binomial Distribution. We derive an iterative approach based on the Expectation-Maximization framework to obtain the MAP estimates. This derivation builds on the approach presented by (Minka 2002) for obtaining the MLE of a Negative Binomial Distribution.

There are two nested EM algorithms in the estimation procedure. The inner EM steps estimate the parameters of the Negative Binomial for a distribution over the latent class variables. The outer EM steps estimate the parameters of the HMM. Since at the start of the outer EM algorithm the distribution over the latent classes changes the most there is limited value in iterating the inner EM steps for a long time to obtain a very precise estimation of the parameters of the Negative Binomial distribution. Therefore, at the start of the outer EM we stop early in the iteration of the inner EM algorithm. However, as the HMM converges we progressively run the inner EM steps longer to obtain a more precise estimate of the parameters.

## Prediction

The task of the 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 $t$ 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 $i$ will be observed in $t+1$ can be computed as:

$$P(i \in I_u^{t+1}) = \sum_k P(Z_u^{t+1} = k)\, P(i \in I_u^{t+1}; a_k, b_k, \boldsymbol{\theta_k}) \tag{8}$$

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 Expression (8) is equivalent to their order by the following quantity (Appendix):

$$R(i,u) = -\sum_k P(Z_u^{t+1} = k)(1 + b_k\theta_{ki})^{-a_k} \tag{9}$$

## Comparison with Existing Methods

We compare the proposed dynamic model with three static algorithms and one dynamic collaborative filtering algorithm that has been recently proposed for explicit rating data.

### User-user similarity based collaborative filtering

The algorithms based on the pairwise 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.

$$R(i,u) = \overline{R_u} + \frac{1}{\sum_{v=1}^{|U|} abs(sim(u,v))} \sum_{v=1}^{|U|} sim(u,v)(R_{ij} - \overline{R_u}) \tag{10}$$

The expected rating a target user $u$ would give to item $i$ is computed by sum of ratings of the other users $(v)$ weighted by the similarity of those users to the target user $u$. $\overline{R_u}$ is the average rating of the user. 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 for obtaining baseline results 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 the EM algorithm. For a user $u$ the items can be recommended in the decreasing order of the probability of $u$ selecting a particular item:

$$R(i,u) = P(i|u) = \sum_{Z} P(Z)P(u|Z)P(i|Z) \tag{11}$$

### Matrix factorization

In recent collaborative filtering literature Matrix Factorization based approaches have been shown to perform very well. Therefore, we compare our dynamic collaborative filtering algorithm with a Singular Value Decomposition based method. Singular value decomposition of a matrix $X$ of size $M \times N$ can be expressed as

$$X \approx U\, S\, V^T \tag{12}$$

where, $U$ is a matrix of size $M \times k$, $V$ is a matrix of size $N \times k$, and $S$ is a diagonal matrix of size $k \times k$. Usually $k \ll M, N$. This results in a more parsimonious representation of $X$. The $k$ pairs of corresponding

columns of $U$ and $V$ represent $k$ latent factors of the matrix $X$. If the matrix consists of the ratings given by $M$ users on $N$ items, then each row of $U$ can consist of a user's affinity to different *categories* of items. Each row of $V^T$ consists of the membership of the item to different categories. A good match between these two rows would lead to a recommendation. There are many ways the similarities between a user record and an item record can be computed. But the approximated matrix $U\,S\,V^T$ provides the similarity scores between the users and items. Therefore, the recommendation score of an item $i$ for a user $u$ can be calculated as

$$R(i,u) = U\,S\,V^T \tag{13}$$

Note that with explicit ratings we often have a large number of ratings missing from the rating matrix. Therefore, special care needs to be taken to perform matrix factorization. However, with implicit ratings we only know whether a user selected an item or not. Therefore, the matrix is taken to be complete (Koren et al. 2009).

### timeSVD++ (Koren 2010)

The *timeSVD++* algorithm is designed for recommending movies to users when explicit ratings are available from the users. It is a matrix factorization approach that has temporal dynamics. This method estimates the ratings as a sum of three distinct parts: time varying item component, time varying user component, and time varying user-factor component. The rating estimation equation is:

$$\widehat{r_{uit}} = \mu + \{b_i + b_i(t)\} + \{b_u + \alpha_u dev_u(t) + b_{ut}\} + \left\{\boldsymbol{q_i^T}\left(\boldsymbol{p_u} + \boldsymbol{\beta_u} dev_u(t) + p_{ktu} + \frac{\sum_{j\in R(u)} y_j}{\sqrt{|R(u)|}}\right)\right\} \tag{14}$$

where, $b_i$ is the average deviation of rating of the item $i$ from the average rating $\mu$ of all items. $b_i(t)$ is a time specific component for item $i$.

$b_u$ is the average deviation of the rating the user $u$ has given to all the items. $dev_u(t)$ is a user specific smooth function of time $t$. Borrowing from (Koren 2010) we use the following formulation of $dev_u(t)$

$$dev_u(t) = sign(t - t_u) \times |t - t_u|^{\beta}$$

where $t_u$ is the average time stamp of the user's ratings. $\alpha_u$ is the user specific weight controlling how much the user's average rating fluctuates. $b_{ut}$ is a parameter to absorb time period specific fluctuation in a user's rating.

The factorization component consists of static item factor weights $\boldsymbol{q_i}$ and a time varying user factor. The user factor again has a component that varies smoothly over time: $\boldsymbol{\beta_u} dev_u(t)$. $p_{ktu}$ is a parameter used to

absorb any variation in the user factor specific to each time period $t$. If there are $K$ factors then $\boldsymbol{q_i}$, $\boldsymbol{p_u}$, and $\boldsymbol{\beta_u}$ are $K \times 1$ vectors. $\boldsymbol{\beta_u}$ is a set of factor specific weights for the user $u$ that determines how much the user factor can vary over time. $R(u)$ is the set of items rated by the user $u$.The variable $y_j$ is an item specific parameter. $\frac{\sum_{j \in R(u)} y_j}{\sqrt{|R(u)|}}$ is a factor designed to reduce the disproportionate amount of influence of the users who have rated a large number of items. In our application scenario, since we have implicit ratings, we treat each user's visit to a blog article as a rating 1 and lack of visit to a blog article as rating 0. Therefore, the set of items $R(u)$ is same for all users. Therefore, the term $\frac{\sum_{j \in R(u)} y_j}{\sqrt{|R(u)|}}$ is the same for each user and reduces to *one* unknown parameter. We denote it as $\gamma$.

The parameters in Equation (14) are estimated by minimizing the sum of squared error in estimating the ratings in the training set and an L2 penalty term to regularize the magnitude of the parameters. Following the original paper we set the regularization parameter to 0.01. However, a sensitivity analysis shows that the results are stable with respect to this parameter.

The *timeSVD++* algorithm was designed to estimate ratings for a time period that was within the range of training data. However, it can be modified to the scenario where one is interested in predicting ratings at a future time period. If there are $T$ time periods in the training data we are interested in making a recommendation for time period $T + 1$. The values of the parameters $b_i(t), b_{ut}$, and $p_{ktu}$ are unavailable for time period $T + 1$. Therefore, to use the prediction Equation (14) we have two options:

1. Use the values of these parameters in time period $T$. The argument for this strategy would be that the values of the parameters in time period $T + 1$ would be most similar to the parameter values in time period $T$.
2. Do not use these transient parameters for prediction of ratings in time period $T + 1$. Instead, only use the persistent parameters for prediction. The argument for this strategy is that the time period specific parameters absorb any period-to-period fluctuation that cannot be explained by any systematic structure. Thus the values of these parameters are errors and are not useful for making a prediction for time period $T + 1$. Thus the prediction equation would look like:

$$\widehat{r_{uit}} = \mu + b_i + b_u + \alpha_u dev_u(t) + \boldsymbol{q_i^T}(\boldsymbol{p_u} + \boldsymbol{\beta_u} dev_u(t) + \gamma)$$

   We find from our experiments that this strategy produces better results. Therefore, we use the second strategy to generate recommendations for time period $T + 1$.

Using the convention used for the User-user similarity algorithm and the matrix factorization based algorithm we code the reading of an article by a user in a time period as a rating of 1 and lack of reading as a rating of 0.

## Experiments

Each algorithm is trained on data up to time period $t$ and is used to predict what each user will read in period $t + 1$. Using the standard convention in the literature the dataset 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. We first present the performances of the algorithms by setting both these thresholds at 400. Later we present the results of sensitivity analysis where the threshold is varied.

### Model Selection

During the training of model based algorithms the probability of training data can be arbitrarily increased by increasing the number of latent classes or factors. However, it leads to over fitting and does not improve the predictive performance of the algorithm on the test dataset. Bayesian Information Criterion (BIC) is one of the popular criteria to select the number of latent classes that is closest to the true model given the data (Schwarz 1978).

$$\mathrm{BIC}(s) = l_s - \frac{p_s}{2}\log n \tag{15}$$

where $s$ indexes a model with certain number of latent classes, $l_s$ is the maximum likelihood of the model $s$, $p_s$is the number of parameters in the model, and $n$ is the number of data points. This score increases with likelihood of the model, but penalizes the complex models. BIC approach prescribes that one should select the model with the highest BIC score.

Akaike Information Criterion (AIC) is another criterion for selecting the best model (Akaike 1974).

$$\mathrm{AIC}(s) = l_s - p_s \tag{16}$$

AIC penalizes the models less severely for their complexity. Another implicit difference between the two model selection criteria is that BIC attempts to find the true model assuming that it is among the set of models considered; however, AIC does not make any assumption about the true model being in the set of models considered and attempts to maximize the prediction accuracy in the test set. In our experiments we find that BIC penalizes the models too aggressively for the number of latent classes and hurts the prediction performance. On the other hand, AIC seems to provide a more reasonable selection criterion. The AIC scores for the HMM and the static Aspect model are shown in Figure 4. From these plots we see

that the best scores for the static models are obtained by using 4 to 6 latent classes and the best scores of the HMM are obtained using 20 to 30 latent classes. The recommendation accuracies in the next section have been reported using 5 latent classes for the static model and 25 latent classes for the HMM model.

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. However, since the complexity of the EM algorithm used to estimate the parameters of the model is linear with respect to the size of the dataset we are able to complete the task relatively quickly (within 10—20 minutes using commodity hardware). However, for richer dynamic model where deterministic approximations such as the EM are not available and we must resort to stochastic approximation methods such as the MCMC sampling, it might be difficult to explore large number of classes.

The SVD based matrix factorization algorithms also require one to choose the number of components to use. Scree plots of singular values are commonly used as a guideline to select the number of components to use. Scree plots of the training data sets are shown in Figure 5. From the figure we see that most of the variances of the training data can be explained using 4 to 8 components. In the reported results we have set the number of components to 4 since it produces the best performance in the test dataset.

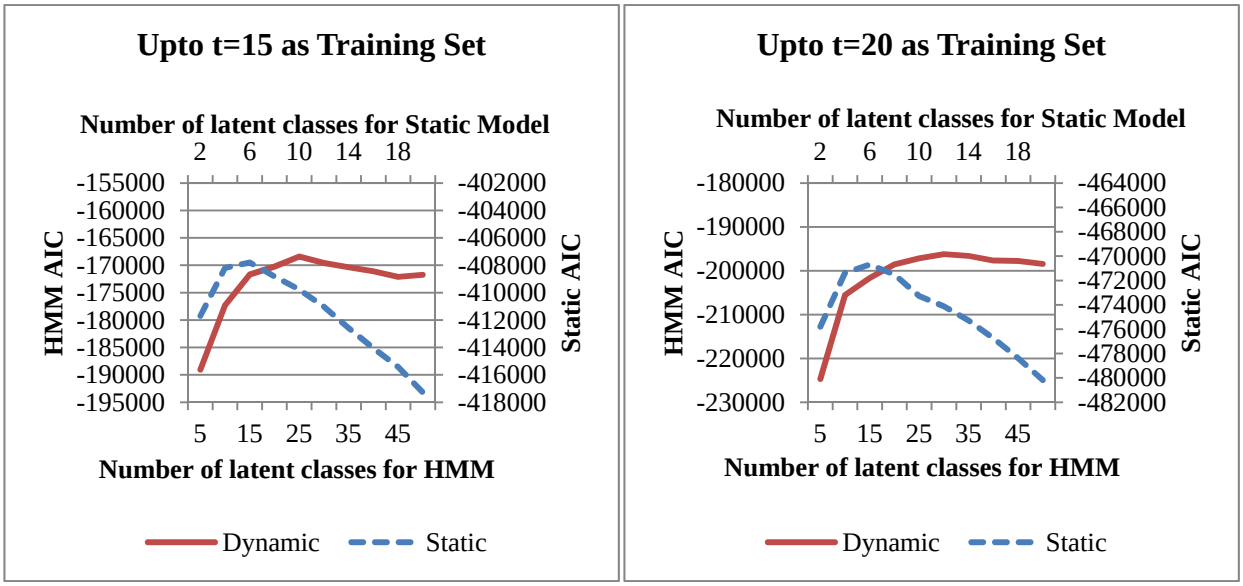


**Figure 4 AIC scores for different number of classes**

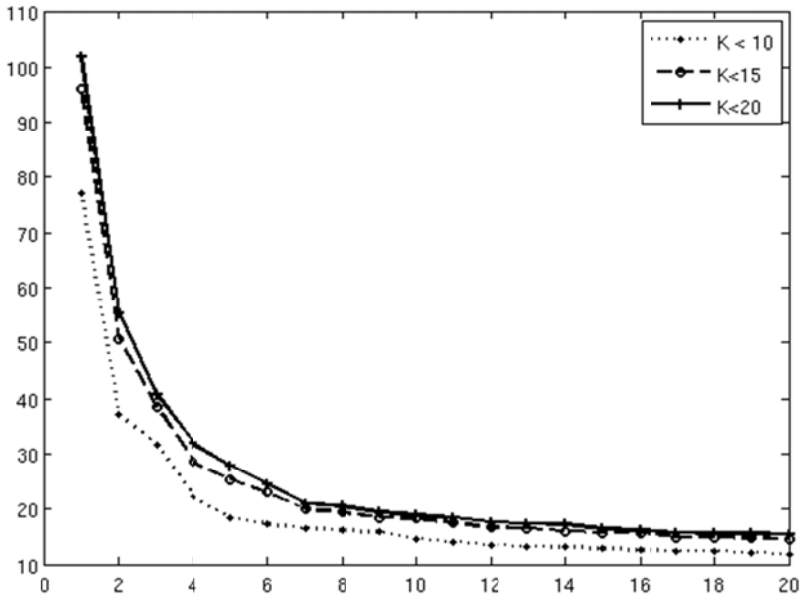


**Figure 5 Scree plot to select the number of components of the SVD**

## Recommendation Performance

The performances of the four recommender systems are measured by their *precision* and *recall* scores. Each algorithm is used to calculate the recommendation score of all the 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 both the numbers at any retrieval level (Herlocker et al. 2004).

$$\frac{1}{F} = \frac{1}{2}\left(\frac{1}{P} + \frac{1}{R}\right) \tag{17}$$

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 out performs all the static algorithms we have tested as well as the *timeSVD++* algorithm for dynamic collaborative filtering algorithm that was designed for explicit ratings.

| | Top 5 | | | Top 10 | | |
|---|---|---|---|---|---|---|
| | *P* | *R* | *F* | *P* | *R* | *F* |
| Dynamic Model | 0.0672 | 0.1721 | 0.0963 | 0.0548 | 0.2598 | 0.0902 |
| Aspect Model | 0.0391 | 0.0700 | 0.0498 | 0.0370 | 0.1273 | 0.0570 |
| User-User Similarity | 0.0378 | 0.0712 | 0.0491 | 0.0313 | 0.1132 | 0.0488 |
| Singular Value Decomposition | 0.0388 | 0.0610 | 0.0470 | 0.0347 | 0.1050 | 0.0516 |
| timeSVD++ | 0.0092 | 0.0147 | 0.0111 | 0.0109 | 0.0316 | 0.0160 |

**Table 1 Precision (P), Recall (R), and F-scores of the four algorithms.**

The poor performance of the *timeSVD++* algorithm, which is very successful on Netflix prize dataset, is not surprising in this context. The algorithm is designed to identify trends in user factors from explicit ratings. However, in this case we apply the algorithm by converting implicit ratings to 0/1 values. The performance of the algorithm on this task suggests that naïve application of the algorithm that was designed for explicit ratings, to the dataset with implicit ratings is not very fruitful.

For each train-test split each algorithm produces an ordered list of items. Precision and Recall measures at 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 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 non-relevant items recommended (False Positive or FP) and the fraction of relevant items that are recommended (True Positive Rate or TP). Thus, for each item in the list we generate a pair of numbers (FP, TP), which are used as the X and Y coordinates, respectively, to plot a set of points. It is easy to note that when there are no items recommended all the methods will produce (FP=0, TP=0). As more items are recommended both these numbers monotonically increase. When all the items are recommended from the list every algorithm would have (FP=1, TP=1). A perfect recommender system would retrieve all 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 the True Positive Rates are extracted from each ROC curve at predetermined False Positive Rates and averaged. The AUCs for all 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 6.

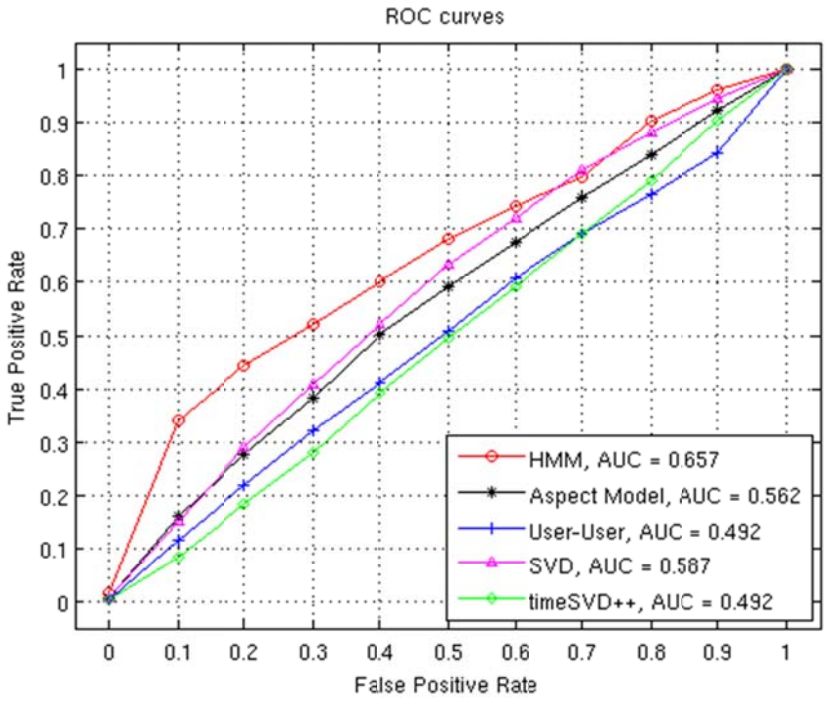


**Figure 6 Average ROC curves and AUC values of each algorithm**

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 2004). 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% level. However, the confidence bands are omitted from Figure 6 for the sake of clarity.

## Latent Classes and Transitions

Upon examination of the transition probability matrix we find that, although there are classes in which the users tend to stay if they are already there, 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 7.

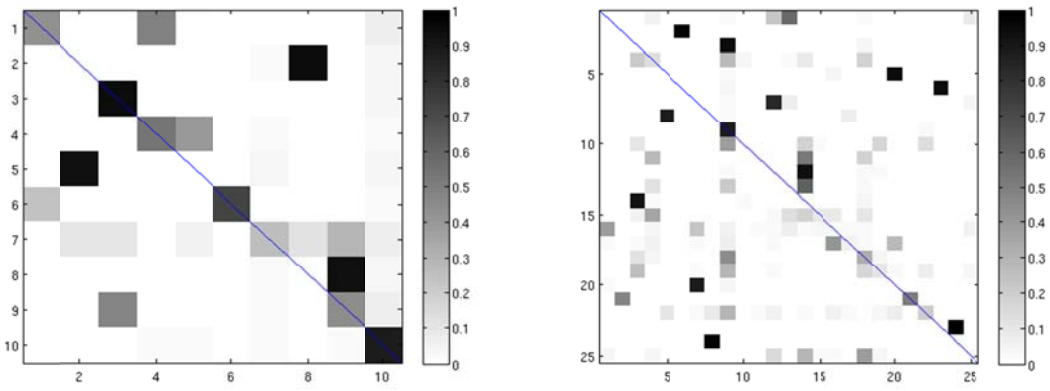

**Figure 7: Transition probability matrix shown as gray scale images for HMM with 10 latent classes (left) and 25 latent classes (right). The darker the cell color is, the larger the transition probability. Although, there are a few large probabilities on the diagonal, there are several large off diagonal probabilities suggesting a class switching behavior for the user.**

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 2 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 are largely different in their selection of favorite articles to read.

| 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 |

**Table 2. 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.**

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. It is improbable that the users will stay in a highly active class for long. On the other hand, class 10 has little activity, and the users who are in class 10 tend to stay in class 10.

## Sensitivity Analysis

The results obtained so far are 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. This threshold of user and article selection controls the density of the data. In addition all results were obtained by using 25 latent classes in the HMM, 5 latent classes for the Aspect model, and 4 factors in the SVD based algorithm. These were the models that produced the best results for each method on this particular subset of data. We now vary the density of the dataset and evaluate the methods over a range of latent classes. We found that the performance of the HMM generally improves with the number of latent classes; however, the performances of the static methods are the best when 5—20 classes are used. This is also observed in the AIC scores of the algorithms (Figure 4). Therefore, we evaluate the dynamic model by varying the number of classes over 10—100. But, we evaluate the static methods by varying the number of latent classes from 2—20.

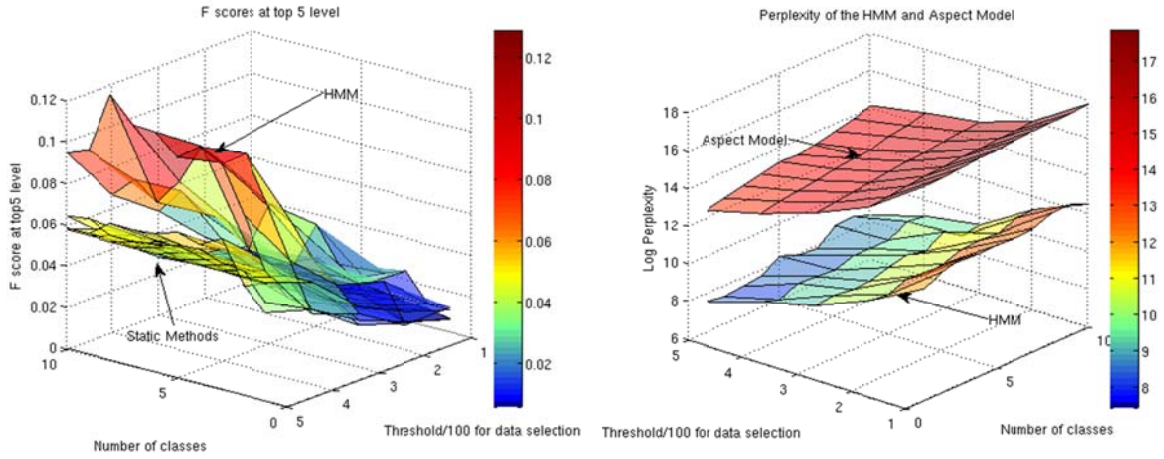


**Figure 8. Sensitivity analysis is performed by varying the data densities and the number of classes and factors. The number of classes used for HMM was varied from 10—100 and from 2—20 for the static models. The data density was controlled by varying the minimum number of articles a user must visit and the minimum number of users an article must be read by from 100—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 for all four methods. The *Perplexity* of the HMM and Aspect Model are compared in the right plot.**

In addition to the F score of the algorithms at the top-5 level, we compare the perplexity of the two model based algorithms on the test set. The perplexity of a probabilistic model is defined as the $e^{-\overline{ll_{test}}}$. A lower

log likelihood of the model on the test data would lead to higher perplexity. Thus, perplexity is a measure of 'surprise' of the probability model upon seeing the test data. It is often used in Information Retrieval and language modeling literature to measure the quality of a probabilistic model in predicting future events, e.g., word occurrences in a set of previously unseen documents (Azzopardi et al. 2003).

The results of these sensitivity analyses confirm our earlier findings. The HMM has a higher F-score across the data densities and the number of classes (Figure 8, left plot). In addition we observe that the performance of the HMM improves with increasing data density due to increased availability of the training data. The performance of the HMM is stable and best when 25—50 classes are used. However, it shows signs of over fitting when more than 50 classes are used.

The perplexity comparison is also interesting (Figure 8, right plot). The HMM is less surprised by the test data than the static Aspect model is. The perplexity of each method decreases as more training data is available for each user. One effect of decreasing the selection threshold for each user is that we have more users and items in the dataset, although it reduces the number of observations per user on average. When there are more users and items, increasing the number of classes does not increase the perplexity; however, when there are only a few users and items, increasing the number of classes leads to over fitting and increases the perplexity of the two models.

# Conclusion

## Summary

We present a hidden Markov model for Collaborative Filtering of implicit ratings that accounts for changing user preferences. Implicit ratings are much more common than explicit ratings. Therefore, the collaborative filtering algorithms for implicit ratings have wider application. Despite evidence from the literature that a user's preference changes 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 user's preferences are changing. These include the uncertainty that the user preference might be different during the test period than they were in the training period, uncertainty about which of many possible preferences generated the data at any time period, etc. In addition one usually does not observe multiple ratings from a user on an item. Therefore, we face the challenge of learning changes in a user's preference from their rating on distinct items over time. In summary, the task of a static collaborative filtering algorithm is to infer stable global

preference patterns and where each user lies in that space. With dynamic collaborative filtering the challenge is to infer stable global preference patterns and how each user is moving in that space.

We propose an HMM to address these challenges. The preference of each user is represented as degree of memberships in a set of latent classes. Each latent class represents a global preference pattern that governs the amount of articles visited in each month and the selection of those articles. This dual purpose of the latent classes requires an innovation in the observation model of the HMM. We represent the observation model as a Negative Binomial Mixture of Multinomial distribution and derive an estimation procedure that scales to large datasets.

We evaluate the proposed algorithm on a dataset collected from a large IT services firm over 22 months. The dataset consists of time stamped record of employees' visits to blog articles posted on the corporate blog network. We find that the proposed algorithm outperforms popular model based, instance based, and matrix factorization based static collaborative algorithms found in the literature. In addition, it significantly outperforms the matrix factorization based algorithm that has been proposed recently for dynamic collaborative filtering of explicit rating datasets. The performance advantage of the dynamic model is stable and significant across different data sparsity levels and across the number of classes used in the model.

## Implications

Improved recommendation performance is important. 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 preference one can make more effective recommendations.

## 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.

In the presented work we model the changes to the user preference by a Markov model, by which users' preference in the next time period probabilistically depends only on the user's preference in the current time period. A model that uses more of a user's preference history can be explored. (Koren 2010) models

changes in users' average explicit ratings on items using linear and polynomial trend functions. However, modeling implicit ratings using such trend functions is an open research question.

We assume one global transition probability matrix for all users. This is a strong assumption. Users can differ in which preference behavior they switch to and how long they might take to make the switch. Therefore, incorporating user level heterogeneity in the transition behavior can lead to further improvements in the recommendation. The common strategies used to estimate models with user heterogeneity involve use of stochastic approximation via the MCMC sampling. However, recommender system applications require more scalable approaches. Deterministic approximation based on a Variational-Bayesian framework is an alternative approach with some nice properties such as well defined bounds on the approximation quality, scalability, etc.

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, researchers have struggled to show advantages of such additional data (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.

## Bibliography


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*), pp 734-749.

Adomavicius, G., and Tuzhilin, A. 2010. "Context-Aware Recommender Systems," in: *Recommender Systems Handbook: A Complete Guide for Research Scientists and Practitioners*. Springer, pp. 335–336.

Akaike, H. 1974. "A New Look at the Statistical Model Identification," *Automatic Control, IEEE Transactions on* (19:6), pp 716-723.

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

Azzopardi, L., Girolami, M., and van Risjbergen, K. 2003. "Investigating the Relationship between Language Model Perplexity and Ir Precision-Recall Measures," ACM, pp. 369-370.

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," *KDDCup*: Citeseer.

Billsus, D., and Pazzani, M.J. 1998. "Learning Collaborative Information Filters," in: *Proceedings of the Fifteenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers.

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

Breese, J.S., Heckerman, D., and Kadie, C. 1998. "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," Microsoft Research.

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

Cerra, V., and Saxena, S. 2005. *Growth Dynamics: The Myth of Economic Recovery*. International Monetary Fund.

Chen, A. 2005. "Context-Aware Collaborative Filtering System: Predicting the User's Preference in the Ubiquitous Computing Environment," *Location-and Context-Awareness*), pp 244-253.

Chien, Y.H., and George, E.I. 1999. "A Bayesian Model for Collaborative Filtering," *Proceedings of the 7 thInternational Workshop on Artificial Intelligence and Statistics*).

Cunningham, P., Nowlan, N., Delany, S., and Haahr, M. 2003. "A Case-Based Approach to Spam Filtering That Can Track Concept Drift," Citeseer, pp. 2003-2016.

Cutting, D., Kupiec, J., Pedersen, J., and Sibun, P. 1992. "A Practical Part-of-Speech Tagger," Association for Computational Linguistics, pp. 133-140.

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*. Banff, Alberta, Canada: ACM, 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), 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.

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," *Workshop on Web Usage Analysis and User Profiling (WEBKDD'99)*).

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

Hamilton, J. 1988. "Rational-Expectations Econometric Analysis of Changes in Regime:: An Investigation of the Term Structure of Interest Rates," *Journal of Economic Dynamics and Control* (12:2-3), pp 385-423.

Hamilton, J. 2005. "What's Real About the Business Cycle?," *NBER Working Paper*).

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," ACM, pp. 230-237.

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

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

Hofmann, T., and Puzicha, J. 1999. "Latent Class Models for Collaborative Filtering," in: *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99-Vol2)*. S.F.: Morgan Kaufmann Publishers, pp. 688-693.

Hu, Y., Koren, Y., and Volinsky, C. 2009. "Collaborative Filtering for Implicit Feedback Datasets," IEEE, pp. 263-272.

Huang, Z., Zeng, D.D., and Chen, H. 2007. "Analyzing Consumer-Product Graphs: Empirical Findings and Applications in Recommender Systems," *MANAGEMENT SCIENCE* (53:7), July 1, 2007, pp 1146-1164.

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

Jeanne, O., and Masson, P. 2000. "Currency Crises, Sunspots and Markov-Switching Regimes," *Journal of International Economics* (50:2), pp 327-350.

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," *Cryptographic Hardware and Embedded Systems-CHES 2003*), pp 17-34.

Kevin, M. 2002. "Dynamic Bayesian Networks: Representation, Inference and Learning." PhD thesis, University of California, Berkley, USA www. ai. mit. edu/~ murphyk/Thesis/thesis. pdf.

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

Koren, Y. 2010. "Collaborative Filtering with Temporal Dynamics," *Commun. 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.

Kschischang, F., Frey, B., and Loeliger, H. 2001. "Factor Graphs and the Sum-Product Algorithm," *IEEE Transactions on information theory* (47:2), pp 498-519.

Lang, K. 1995. "Newsweeder: Learning to Filter Netnews," Citeseer.

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

Lukashin, A., and Borodovsky, M. 1998. "Genemark. Hmm: New Solutions for Gene Finding," *Nucleic Acids Research* (26:4), p 1107.

Macskassy, S., and Provost, F. 2004. "Confidence Bands for Roc Curves: Methods and an Empirical Study," Citeseer, pp. 61–70.

Minka, T. 2002. "Estimating a Gamma Distribution,").

Mooney, R.J., and Roy, L. 2000. "Content-Based Book Recommending Using Learning for Text Categorization," in: *Proceedings of the fifth ACM conference on Digital libraries*. San Antonio, Texas, United States: ACM, pp. 195-204.

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

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

Ostendorf, M., Digalakis, V., and Kimball, O. 1996. "From Hmm's to Segment Models: A Uni Ed View of Stochastic Modeling for Speech Recognition," *IEEE Trans. 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: ACM, pp. 667-676.

Paterek, A. 2007. "Improving Regularized Singular Value Decomposition for Collaborative Filtering," Citeseer.

Pazzani, M., Muramatsu, J., and Billsus, D. 1996. "Syskill & Webert: Identifying Interesting Web Sites."

Pilászy, I., and Tikk, D. 2009. "Recommending New Movies: Even a Few Ratings Are More Valuable Than Metadata," ACM, 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.

Rendle, S., Freudenthaler, C., and Schmidt-Thieme, L. 2010. "Factorizing Personalized Markov Chains for Next-Basket Recommendation," ACM, pp. 811-820.
Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. 1994a. "Grouplens: An Open Architecture for Collaborative Filtering of Netnews," ACM, pp. 175-186.
Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. 1994b. "Grouplens: An Open Architecture for Collaborative Filtering of Netnews," in: *Proceedings of the Conference on Computer-Supported Cooperative Work, CSCW'94*.
Resnick, P., and Varian, H.R. 1997. "Recommender Systems," *Commun. ACM* (40:3), pp 56-58.
Sahoo, N., Krishnan, R., Duncan, G., and Callan, J. 2010. "The Halo Effect in Multi-Component Ratings and Its Implications for Recommender Systems: The Case of Yahoo! Movies," *Information Systems Research*).
Sarwar, B., Karypis, G., Konstan, J., and Reidl, J. 2001. "Item-Based Collaborative Filtering Recommendation Algorithms," ACM, p. 295.
Schlimmer, J., and Granger, R. 1986. "Beyond Incremental Processing: Tracking Concept Drift," pp. 502–507.
Schwarz, G. 1978. "Estimating the Dimension of a Model," *The annals of statistics* (6:2), pp 461-464.
Shardanand, U., and Maes, P. 1995. "Social Information Filtering: Algorithms for Automating \"Word of Mouth\"," in: *CHI*. pp. 210-217.
Si, L., and Jin, R. 2003. "Flexible Mixture Model for Collaborative Filtering," in: *ICML*. AAAI Press, pp. 704-711.
Sims, C., and Zha, T. 2006. "Were There Regime Switches in Us Monetary Policy?," *The American Economic Review* (96:1), pp 54-81.
Singh, P., Youn, N., and Tan, Y. 2006. "Developer Learning Dynamics in Open Source Software Projects: A Hidden Markov Model Analysis." Citeseer.
Singh, P.V., Sahoo, N., and Mukhopadhyay, T. 2010. "Seeking Variety: A Dynamic Model of Employee Blog Reading Behavior."
Street, W., and Kim, Y. 2001. "A Streaming Ensemble Algorithm (Sea) for Large-Scale Classification," ACM, pp. 377-382.
Swets, J.A. 1963. "Information Retrieval Systems," *Science* (141:3577), July 19, 1963, pp 245-250.
Tsymbal, A. 2004. "The Problem of Concept Drift: Definitions and Related Work," *Computer Science Department, Trinity College Dublin*).
Ungar, L.H., and Foster, D.P. 1998. "Clustering Methods for Collaborative Filtering," *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," 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.