Adapting Recommendations to Contextual Changes Using Hierarchical Hidden Markov Models

Mehdi Hosseinzadeh Aghdam
DePaul University
Chicago, IL 60604, USA
maghdam@depaul.edu

Negar Hariri
DePaul University
Chicago, IL 60604, USA
nhariri@cs.depaul.edu

Bamshad Mobasher
DePaul University
Chicago, IL 60604, USA
mobasher@cs.depaul.edu

Robin Burke
DePaul University
Chicago, IL 60604, USA
rburke@cs.depaul.edu

ABSTRACT

Recommender systems help users find items of interest by tailoring their recommendations to users' personal preferences. The utility of an item for a user, however, may vary greatly depending on that user's specific situation or the context in which the item is used. Without considering these changes in preferences, the recommendations may match the general preferences of a user, but they may have small value for the user in his/her current situation. In this paper, we introduce a hierarchical hidden Markov model for capturing changes in user's preferences. Using a user's feedback sequence on items, we model the user as a hierarchical hidden Markov process and the current context of the user as a hidden variable in this model. For a given user, our model is used to infer the maximum likelihood sequence of transitions between contextual states and to predict the probability distribution for the context of the next action. The predicted context is then used to generate recommendations. Our evaluation results using Last.fm music playlist data, indicate that this approach achieves significantly better performance in terms of accuracy and diversity compared to baseline methods.

1. INTRODUCTION

Traditional recommender systems recommend items based on users' history of preferences and interactions. In general, however, users' preferences can change because of the natural evolution of their tastes, changes in their current context, or the emergence of new items. As a result of these preference changes, a static set of recommendations generated based on the full set of past preferences does not always provide the most utility for a user. In particular, in domains where there may be frequent changes in user's context during the course of interaction with the system, failure to consider these changes may result in considerable performance degradation. For example, the user of a music recommender may have different interests in music depending on her/his current activity (such as exercising, relaxing, driving, etc.).

To address this issue context-aware recommender systems (CARSs) have been introduced [1]. Compared to the traditional systems that mainly utilize users' preference history, CARSs provide more relevant results by generating recommendations that match the current interests of each user. In some applications, the current context of the user might be explicitly given to the system as a set of factors affecting the recommendations. On the other hand, if the contextual factors explaining the preference changes are not directly observed, it might still be possible to infer them as a set of latent variables and use them to generate recommendations during the course of a user's interactions with the system.

In this paper, we focus on a setting where users' feedback on items is revealed sequentially. Based on the feedback data in the current session and also the user's history of preferences, the recommender predicts the next item that would be interesting for the user. For example, based on the sequence of songs that a user has listened to in her current playing session, the recommender suggests other songs that might be of interest.

We propose a context-aware recommendation algorithm based on hierarchical hidden Markov modeling (HHMM) [4]. Our model captures common patterns of contextual changes in users' preferences, represented as hidden variables in the model, and use them to produce personalized recommendations matching the current interests of the user. Also, the proposed approach increases diversity of recommendations while maintaining an acceptable level of accuracy. The ability to capture user preferences at a more aggregate level of latent variables representing contexts, enables our approach to produce more diverse recommendation lists.

We evaluated our approach using an off-line cross validation strategy. In our experiments we used Last.fm dataset containing time-stamped sequences of users' listening activities. Our evaluation results indicate that our proposed approach achieves significantly better performance in terms of accuracy and diversity of the recommendations compare to other baseline methods.

RecSys'15, September 16–20, 2015, Vienna, Austria.

DOI: http://dx.doi.org/10.1145/2792838.2799684.

2. RELATED WORK

Many of the existing conventional recommendation methods, such as matrix factorization or neighborhood-based methods, don't take into account the users' change of interests. Time-aware recommendation approaches such as [6] consider the drift in users' preferences over time and model the temporal dynamics within the data. However, these methods assume that time is the only factor that can impact users' interests while ignoring other possible contextual factors.

Recommendation based on pattern-mining [7][5], can be one possible approach for context-adaptation in sequential recommenders. This technique relies on discovering frequent patterns of items based on sequence of items in the training data and using these patterns to generate recommendations.

Recommendation based on a Markov chain model [8] is another method that can utilize such sequential data by predicting users' next actions based on their previous preferences. For an m^{th} order model, the recommendations are generated based on only the last m recent observations from each user. So, if m is a small value, then the recommender is not generating personalized recommendations as the algorithm is ignoring a large part of the information in a user's profile. On the other hand, setting m to a large value increases the number of model parameters exponentially which makes it impractical to learn the model in many situations.

Another alternative is the hidden Markov modeling (HMM) approach [11], where the states are latent variables and the items liked by the user in that state are observed variables. Each state has a probability distribution over the set of items. The learning process includes estimating the transition probabilities between the states as well as the probability distribution of observing items at each hidden state. The distribution of current state changes as result of changes in a user's behavior. Therefore, these latent states can be used as a representation of the current context of the user.

Many of the recommendation algorithms are biased toward recommending a few popular items. These algorithms often can achieve good precision and recall levels in comparison to more complex methods. In fact, according to the analysis in [3], the recommendation accuracy captured based on precision and recall can be increased when an additional popularity bias is introduced. However, it has been shown that these popular items are not extremely exciting for the users and do not help with increasing the sales revenue as much as more niche and new items. Similar studies (e.g., [2]) have shown that recommendation in the long tail has special value in increasing the sales revenue and producing more useful and diverse recommendations.

3. HIERARCHICAL HIDDEN MARKOV MODEL

In this paper, we focus on a setting where users' preferences are revealed as a time-stamped sequence of positive feedback on items, such as playing songs or clicking on Web pages. The goal of the recommender is to predict the future interests of the user. We introduce a context-aware recommender that models the changing contextual states of the user based on the feedback sequence collected from that user. The recommender system monitors variations in users' preferences and dynamically adapts to these changes.

Our approach is based on training an HHMM for modeling context as a latent variable affecting the users' feedback behaviors. Hierarchical hidden Markov model is a type of structured hierarchical stochastic process. HHMM extends the original hidden Markov model by training two levels of hidden variables. The hidden variables at the first level are used as observations for training the second level.

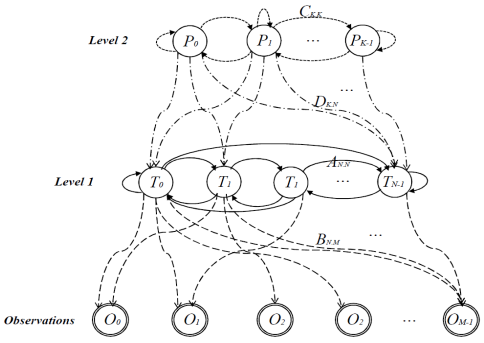

Figure 1: Graphical model of HHMM.

Table 1: Model Parameters

N = Number of states in the first level.
M = Number of items.
A = State transition probabilities in the first level.
B= Observation probability matrix between the first level of hidden states and items.
C = State transition probabilities in the second level.
D = Observation probability matrix between second and first levels.
π = Initial state distribution.
$O =< O_0, O_1, \cdots, O_{L-1} >$ = observation sequence.
L =Length of the observation sequence.

In our system, we used users' sequences of positive feedback on items as the set of observations to train the first level of hidden states. The first level of hidden variables, represent our model of the latent contextual states affecting users' preferences over time. The second level of hidden variables, captures the common patterns of transition between different contextual states and correspond to our latent model of users' interests. Figure 1 presents the overall process of HHMM in our system. Table 1 describes the notation used in the presented graphical model.

3.1 Inference and Recommendation

Algorithm 1 presents the general process of training the HHMM in our system and using it for producing context-aware recommendations [4]. The model parameters that need to be inferred are shown as $\lambda(A, B, C, D, \pi)$ and are described in table 1. At the beginning, each of these parameters is randomly initialized. Let $O =< o_1, o_2, \cdots, o_l >$ represent a feedback sequence with size l where o_i indicates the i^{th} item in the sequence that received positive feedback from the user. Given each user feedback sequence, parameters A and B are first updated. In other words, for each input sequence, the state distribution, the first-level state

Algorithm 1: HHMM for context-aware recommendation

```
1- Initialize λ = (A, B, C, D, π) to small random values.
2- For each user:
3-     For each user's feedback sequence:
4-         Re-estimate λ(A, B, π).
5-         Find the most likely first-level state sequence.
6-         Re-estimate λ(C, D, π) based on first-level state
           sequence inferred in the previous step.
7-         Find the most likely second-level state sequence
           based on first-level sequence.
8-         Predict next context using λ(C, D, π).
9-         Predict next observation using λ(A, B, C, D, π).
```

transition probabilities, and the observation probability matrix (between the first-level latent variables and the items) are updated. These parameters are updated as follows:

for *i=0..N-1* **do**
  $\pi_i = \gamma_0(i)$
for *i=0..N-1* **do**
  for *j=0..N-1* **do**
    $a_{ij} = \sum_{l=0}^{L-2} \gamma_l(i,j) / \sum_{l=0}^{L-2} \gamma_l(i)$
for *j=0..N-1* **do**
  for *k=0..M-1* **do**
    $b_j(k) = \sum_{l \in \{0,\ldots,L-2\}\ ,\ o_l=k} \gamma_l(j) / \sum_{l=0}^{L-2} \gamma_l(j)$

Where $\gamma_l(i,j) = P(x_l = s_i, x_{l+1} = s_j | O, \lambda)$ is the transition probability from state s_i to state s_j at time $l+1$, and $\gamma_l(i) = \sum_{j=0}^{N-1} \gamma_l(i,j)$. In the next step, the first-level state sequence with maximum likelihood is discovered. This corresponds to the most likely transition sequence between contextual states that explains the user feedback sequence. We applied Viterbi algorithm to solve this problem [10].

The first-level state sequence discovered in the previous step is then used to update matrices C and D. The procedure we used in this step is similar to the method used for updating the matrices A and B. The only difference is in the type of input: the parameters C and D are updated based on the discovered first-level state sequences while A and B are re-estimated based on the observed item sequences.

In the next step, the second-level state sequence with maximum likelihood is determined. This sequence is the most likely transition sequence between the second-level hidden variables that explains the contextual changes. Based on the trained context transition probabilities (parameters C and D), our model predicts the context of the next action of the user. By using the predicted context and having the observation probability matrix B, our model then generates the recommendations that match the predicted context. These recommendations are chosen as the set of items that have the highest probability given the predicted context.

After training the model, we used the multiplication of C and D to predict the next context for a given user. In the next step, by multiplication A and B, we compute the probability of each item for the recommendation based on the contexts. Finally, by using these probabilities and having predicted context, we predict the top-N recommendations with the highest probability that correspond to predicted context.

4. EXPERIMENTS

This section describes our experiments for evaluating our sequential recommendation approach against a number of other popular baseline methods. We did our experiments based on users' music listening activities collected from Last.fm Website between 2009/01/01 to 2009/05/31. This dataset contained the time-stamped sequence of artists that each user had listened to during the specified time period. We used the first four months of the data for training the models and the last month (2009/05) for evaluation of the recommendations. The training set contained 837 users, with at least one artist in the test and train partitions, and the number of unique artists was 51759. The test data contained 462 users. Given a user, each of the competing algorithms produced a set of N recommendations which were then evaluated based on the test data.

4.1 Evaluation Metrics

All the recommendation methods were compared based on the *precision* and *recall* metrics. These two measures are among the common metrics used for off-line evaluation of recommendation algorithms where binary type of feedback is available. Given a set of recommendations, let TP represent the number of correct recommendations, and FP indicate the number of incorrect recommendations. A recommendation is assumed to be correct only if it appears in the test data. The precision and recall are computed as:

$$Precision = \frac{TP}{TP+FP}, Recall = \frac{TP}{TP+FN} \quad (1)$$

Where FN indicates the number of relevant items that are not included in the recommendation list. Another metric used in our experiments was F-measure that is defined as:

$$F - Measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (2)$$

Popularity Bias was used in our evaluations to compare the diversity of different recommendation methods. We followed the same approach as in [3] to compute this metric. The items were first sorted based on their overall frequencies in all users' profiles and were then grouped into $I = 10$ bins such that the items in the same bin were similar in popularity. For each baseline method, the top $N = 10$ recommendations generated for each test case were then analyzed to compute the normalized distribution of each category.

4.2 Baseline Recommendation Algorithms

This section focuses on comparing the performance of HHMM against other baseline approaches. We used AIC criterion (AIC=$ML_\lambda - P_\lambda$) to determine the optimal number of latent states for HMM and HHMM. Where ML_λ is the maximum likelihood of the model, and P_λ is the number of parameters in the model. Based on this criterion, the number of latent states in the first level was set to 20 and the number of states in the second level was set to 10. The baseline methods used in our evaluations are as followed:

HMM: Recommendation algorithm based on the standard HMM model [11]. Similar to HHMM, we used 20 hidden states for training the HMM model.

Sequential Pattern Mining: Recommendation based on contiguous sequential patterns [7][5] was used as one of the baselines in our evaluations. In our evaluations, the support value of sequential patterns was tuned and set to 0.01.

Item-based Markov Modeling: As another baseline sequential recommender, a k^{th}-order Markov model was built where states corresponded to all item subsequences of length k observed in the training data, and actions corresponded to different items. For each user, the last k items were used to produce recommendations. The transition probability for state s_i and action a_j was computed based on the training sequence database and as the normalized frequency of times a_j occurred after s_j. The order of the model was to $k = 3$.

User-Based kNN: This method was used as one of the baseline approaches. The cosine metric was used to compute the similarity of users and the size of neighborhood for each user was set to 100.

Table 2: Recall and precision for various recommendation methods.

	Top 5			Top 10		
Method	Precision	Recall	F-Measure	Precision	Recall	F-Measure
HHMM	0.3272	0.00655	0.01284	0.27995	0.01125	0.02163
HMM	0.3644	0.0054	0.01064	0.3228	0.0102	0.01977
Sequential Pattern Mining	0.1036	3.56E-05	7.12E-05	0.0812	3.63E-05	7.26E-05
Item-based Markov Modeling	0.0342	8.69E-05	1.73E-04	0.0281	1.23E-04	2.45E-04
User-Based kNN	0.0350	0.0035	6.36E-03	0.0297	0.007	1.13E-02
BPRMF	0.0273	0.0047	8.02E-03	0.0247	0.006	9.65E-03
Most Popular	0.02188	0.006	9.42E-03	0.0179	0.008	1.11E-02
Random	0.0004	5.74E-06	1.13E-05	0.0004	2.76E-05	5.16E-05

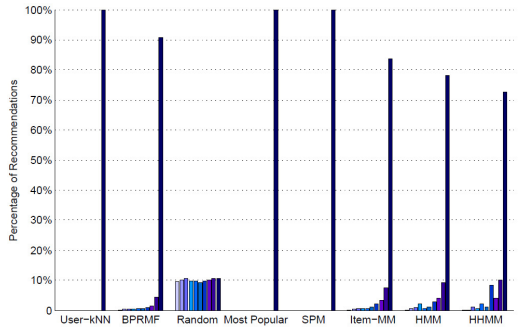

Figure 2: Comparison of popularity bias of different recommendation methods

BPRMF: Matrix Factorization using Bayesian Personalized Ranking (BPRMF) [9] was used as another baseline in our evaluations. This model was trained for 30 factors.

Most Popular: This baseline ranks the items based on popularity.

Random: This recommender, randomly selects N items as the recommendation list for each given user.

4.3 Evaluation Results

Table 2 presents the recall and precision at ranks 5, and 10 for each of the competing algorithms. According to this table, HMM has slightly better precision in comparison to HHMM and also has a significantly higher precision compare to the rest of the baseline methods. However, HHMM has the highest recall and achieves the best overall F-score.

Figure 2 compares the popularity bias of HHMM and other methods. As expected, the random recommender has the lowest popularity bias as the recommendations are almost uniformly distributed over the 10 bins. On the other hand, user-based kNN, most-popular recommender, and recommendation based on sequential pattern mining have the highest popularity bias. All the recommendations generated by these three algorithms belong to a single bin (most popular items) while the items in the remaining bins are not selected for recommendation. Although BPRMF and Item-based Markov model have better diversity than user-based kNN, both HMM and HHMM achieve better performance with respect to lower popularity bias while HHMM has slightly better diversity compare to HMM.

5. CONCLUSIONS

This paper proposes a recommendation method based on Hierarchical Hidden Markov Models for adapting to users' varying interests as a result of contextual changes. We model context as a latent variable affecting the likelihood that a user likes a given item. Our model captures the probability of transition between different contextual states and uses these probabilities to predict the context of the next interaction of the user with the system. The predicted context is then used to generate the recommendations that are most relevant to the current interests of the user as well as his/her history of preferences. Our evaluation results indicate that using the predicted context can improve the diversity of the recommendations.

6. REFERENCES

[1] G. Adomavicius, B. Mobasher, F. Ricci, and A. Tuzhilin. Context-aware recommender systems. *AI Magazine*, 32(3):67–80, 2011.

[2] O. Celma. *Music Recommendation and Discovery - The Long Tail, Long Fail, and Long Play in the Digital Music Space.* Springer, 2010.

[3] F. G. D. Jannach, L. Lerche and G. Bonnin. What recommenders recommend - an analysis of accuracy, popularity, and sales diversity effects. In *User Modeling, Adaptation, and Personalization*. Springer, 2013.

[4] S. FINE, Y. SINGER, and N. TISHBY. The hierarchical hidden markov model: analysis and applications. *Machine Learning*, 32, 1998.

[5] N. Hariri, B. Mobasher, and R. Burke. Context-aware music recommendation based on latenttopic sequential patterns. In *In Proc. of the sixth ACM conference on Recommender systems*, pages 131–138. ACM, 2012.

[6] Y. Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, 2010.

[7] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Using sequential and non-sequential patterns for predictive web usage mining tasks. In *Proc.of the IEEE International Conference on Data Mining*, 2002.

[8] J. Pitkow and P. Pirolli. Mining longest repeating subsequences to predict world wide web surfing. In *Proc. USENIX Symp. On Internet Technologies and Systems*, page 1, 1999.

[9] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *In Proc. of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*.

[10] M. S. Ryan and G. R. Nudd. The viterbi algorithm. 1993.

[11] N. Sahoo, P. V. Singh, and T. Mukhopadhyay. A hidden markov model for collaborative filtering. *MIS Quarterly*, 36(4):1329–1356, 2012.