Physica A 518 (2019) 89–98

Contents lists available at ScienceDirect

Physica A

journal homepage: www.elsevier.com/locate/physa

Context-aware recommender systems using hierarchical hidden Markov model

Mehdi Hosseinzadeh Aghdam *

Department of Computer Engineering, University of Bonab, Bonab, Iran

HIGHLIGHTS

- Context-aware recommender systems consider changes in user preferences to generate recommendations.
- This paper presents a novel hierarchical hidden Markov model to model users' latent context.
- The latent contexts are composed of unsupervised hidden context patterns.
- The proposed method uses modeled contexts to predict personalized recommendations.

ARTICLE INFO

Article history:
Received 15 July 2018
Received in revised form 29 September 2018
Available online 22 November 2018

Keywords:
Context-aware recommender system
Hidden Markov model
Latent context
Recommender systems

ABSTRACT

Recommender systems often generate recommendations based on user's prior preferences. Users' preferences may change over time due to user mode change or context change, identification of such a change is important for generating personalized recommendations. Many earlier methods have been developed under the assumption that each user has a fixed pattern. Regardless of these changes, the recommendation may not match the user's personal preference and this recommendation will not be useful to the user based on the current context of the user. Context-aware recommender systems deal with this problem by utilizing contextual information that affects user preferences and states. Using contextual information is challenging because it is not always possible to obtain all the contextual information. Also, adding various types of contexts to recommender systems increases its dimensionality and sparsity. This paper presents a novel hierarchical hidden Markov model to identify changes in user's preferences over time by modeling the latent context of users. Using the user-selected items, the proposed method models the user as a hidden Markov process and considers the current context of the user as a hidden variable. The latent contexts are automatically learned for each user utilizing hidden Markov model on the data collected from the user's feedback sequences. The results of the experiments, on the benchmark data sets, show that the proposed model has a better performance compared to other methods.

1. Introduction

Recently, personalized recommendations have grown dramatically to identify the items that are interesting. These methods help the users to find the items among the many items and they are used in large online stores like Amazon.com. Many of the existing conventional recommender systems (RS) assume that user preferences pattern is static and produce recommendations based on a user's history of interactions and preferences [1]. In general, however, users' preferences can

* Correspondence to: Department of Computer Engineering, University of Bonab, Velayat Highway, Bonab, 55517-61167, Iran.
E-mail address: mhaghdam@bonabu.ac.ir.

https://doi.org/10.1016/j.physa.2018.11.037
0378-4371/

change over time due to their taste changes and their context changes. The users' preferences, for frequently selected items such as listening to music and watching movies, can be changed due to the evolution of their tastes. For example, a user of a music recommender system may have different interests in music based on the current context (sports, driving, etc.). These changes challenge the recommender systems that are trained using the users' past choices. Although traditional recommender systems have the ability to identify users' past preferences, if the user's preference changes after the training stage the production of recommendation based on past preferences does not always provide the most utility for a user. Failure to consider these changes can significantly decrease the system performance.

On the other hand, the epidemic of smart-phones and the computing everywhere gives the RSs access to the contextual information of users [2]. Recently, the use of context-aware recommender systems (CARS) that consider user context information has been developed [3]. CARSs use contextual information such as user activity, time and location to understand user context and their impact on user preferences. The importance of the user's context to improve the quality of recommendations has been reported by different researchers [4–9]. The incorporation of the user's context in the recommendation process increases the applications of CARSs through recommendations that are relevant to the current preferences of the user [2,10]. Compared to other recommender systems that use past user preferences, CARSs provide relevant recommendations that are close to the current context of each user.

This paper proposed a novel hierarchical hidden Markov model (HHMM) to extract latent context from the data. The latent contexts are composed of unsupervised hidden context patterns which modeling as states in the proposed approach. The latent contexts are learned by applying hidden Markov model techniques. In this paper, user selected items are considered as a sequence in a given time period. This paper also assumes that the user context is not available by direct factors, and these factors are dynamic and should be inferred from the user interaction with the system. A recommender system that uses both user selections in the current context and the user's previous preferences can provide interest recommendations for the user. For example, based on the sequence of music that the user is listening to in the current context, the recommender system can suggest music that is interesting [11]. To evaluate the proposed model, the Last.fm and Netflix data sets that consist of sequences of time-stamped data are used. The evaluation results show that the proposed model significantly improves performance both in terms of accuracy and the diversity of recommendations compared with other methods.

The rest of this work is organized as follows. Section 2 outlines the related works. Section 3 explains the proposed hierarchical hidden Markov model for context-aware recommender systems. The baseline approaches and computational experiments are described in Section 4. It also includes a brief discussion of the results and finally, the conclusion is offered in the last section.

2. Related works

There are different definitions of context for RSs. One of the definitions that are so used is provided by Abowd et al. [2]. They define context as "any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between user and system". This is a general definition that limits the context to information that distinguishes the situations. A similar definition is reported by Lieberman et al. [12]. The definition is: "context can be considered to be everything that affects computation except the explicit input and output". In addition to the general definitions, there are several specific definitions of the context. For example, the context can be defined by a vector of features such as time and location [13]. There are several types of context for CARSs which is summarized as follows:

Time context: Features such as time of day, season, and holidays have demonstrated that they have an important impact on recommendations in some applications. For example, in the travel recommender system, the season will affect the production of recommendations. The movie recommender system can offer suggestions in accordance with special days such as Christmas or Thanksgiving.

Physical context: This context shows the physical state of the user and his surroundings, such as location and temperature. For example, using the user's location and previously selected music, CARSs can generate recommendations that are very relevant to the user's preferences.

Social context: People around the user can influence the suitability of the recommendations in some applications. For example, users may have different preferences for watching movies when they are single or with their children.

Modal context: This context represents the mindset of the user, the user's goals, user mood, his experiences, and abilities. Knowing the user is looking to buy a gift for his wife or buying for himself will make the recommender systems generate different recommendations.

Adomavicius and Alexander have proposed three methods for using the contextual information in recommender systems: pre-filtering, post-filtering, and modeling [5]. Pre-filtering and post-filtering methods filter the items before and after the recommendation process, but the contextual modeling applies the users' context directly to the process of recommendation. The contextual modeling is challenging because increasing the variety of contextual information will increase the dimension of the data (i.e., users, items, and contexts). Consequently, increasing the dimensions will increase the sparsity, because training needs a triple <user, item, context>.

Various factors are effective in obtaining contextual information and it is not always possible to obtain all the information. Also, the characteristics and structure of context can be change dynamically over time. In some applications, the user's context is given directly to the system by a set of features. If the features explaining the context of the user are not directly

accessible, it is possible to infer them as a set of latent contexts and use them to generate recommendations for users. The explicit context can be explained by experts better than the latent context because it describes the known user positions. However, the incentive to use the latent context is privacy, data access, and usability considerations. Using explicit context for a CARS can increase privacy issues, because the user exact context for the system is specified, which is not the case in latent context. The latent context can be obtained automatically by using learning techniques on existing data (the sequence of selected items).

Pattern mining is one of the possible ways to adapt the user's context to recommend a sequence of items [14]. This method generates recommendations by identifying the patterns in the item sequence in the training data. Although this method generates recommendations based on changes in user preferences, in cases where the number of items is too high or the data is sparse, finding patterns to produce recommendations is impractical.

The Markov chain model can predict the next choice by prior user preferences [15]. In this model, recommendations are generated based on the last sequence of items selected by the user. If the number of items in the sequence is small, this method cannot generate personalized recommendation because it ignores much of the user profile information. On the other hand, if the number of items is high, then the model parameters grow exponentially which makes it impractical to learn the model in most applications.

Another method is the hidden Markov modeling (HMM) [16]. In this model, the states are latent contexts and the items liked by the user are observations. Each hidden state has a probability distribution over the 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 a result of changes in a user's context. Therefore, these latent states can be used as a representation of the current context of the user.

Many of the recommendation methods are biased toward recommending a few popular items. These methods often can achieve good accuracy in comparison to more complex models. In fact, the recommendation accuracy can be increased when an additional popularity bias is introduced. One of the goals of the proposed model is to increase the diversity of recommendations without reducing the accuracy of predicting recommendations.

3. The proposed method

This paper focuses on the preferences of users, which is presented as a time-stamped sequence. This sequence can consist of positive feedback from the user on the list of songs he/she has heard or click on web pages. The recommender's goal is to predict the user's interests in the future. This paper presents a novel context-aware recommender system that models user's context changes based on the feedback sequence collected from that user. The proposed method monitors the change in user preferences and dynamically adapts itself to these changes.

3.1. Model description

In the proposed method each context is modeled as a hidden variable, which is affected by feedback from the user's behavior. The proposed method identifies common patterns of contextual changes in users' preferences and uses them to predict the next context of the user. The proposed HHMM is a kind of stochastic hierarchical process. HHMM is an extension of the original hidden Markov model, which consists of two levels of hidden variables. Hidden variables in the first level are used as observations for training the second level. In this method, the positive feedback sequences of users on the items are used as a set of observations to train hidden variables in the first level. Hidden variables of the first level represent the latent context of the users over time. Second-level hidden variables identify common patterns between different contextual states. The second level variables are equivalent to the hidden interests of users in the proposed model. Fig. 1 shows the graphical model of HHMM.

3.2. Inference and recommendation

The model parameters to be inferred presented as $\lambda(A,B,C,D,\pi)$ and explained in Table 1. All of the parameters randomly initialized. $O=<O_0,O_1,\ldots,O_{L-1}>$ shows the sequence of observations of size L. In this sequence, O_i represents the ith item that received feedback from the user.

Three problems typically arise in recommender systems that use HHMMs:

(1) Computing the likelihood of a sequence: Given HHMM and its parameters $\lambda(A,B,C,D,\pi)$, find the probability of an observation sequence, $P(O|\lambda)$.

(2) Finding the most likely state sequence: Given HHMM, its parameters and a sequence of observation, find the most probable state sequence.

(3) Estimating the parameters of a model: Given an observation sequence and the dimensions of HHMM, find the parameters of the model that maximizes the probability of observation sequence.

Approaches for these problems for HHMM are more involved than for HMM, due to the hierarchical structure and multi-scale properties. Direct computation is generally infeasible for the first problem since it requires about $2L\times N^L$ multiplications. Using the forward algorithm, complexity is reduced.

$$\alpha_l(i) = P(O_0, O_1, \ldots, O_l, x_l = s_i|\lambda) \tag{1}$$

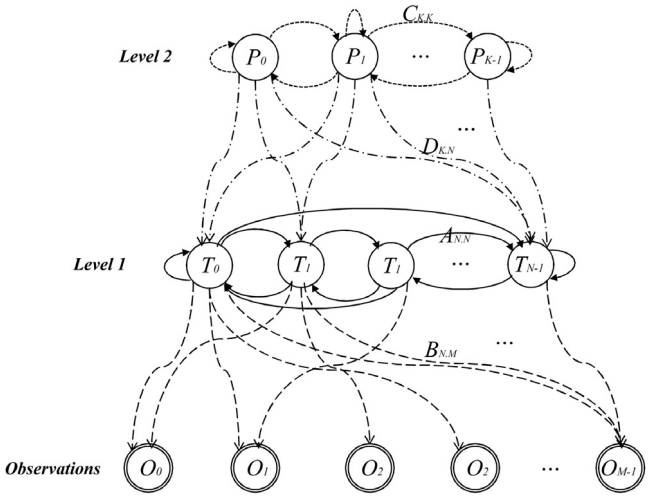

Fig. 1. Graphical model of hierarchical hidden Markov model.

Table 1
The proposed method parameters.

Parameter	Description
N	Number of states in the first level
M	Number of observation symbols (items)
A	State transition probabilities in the first level
B	Observation probability matrix between first levels 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	observation sequence
L	Length of the observation sequence

$\alpha_l(i)$ is the probability of a sequence of partial observations up to l, where underlying Markov process in state s_i at time step l.

$$P(O|\lambda) = \sum_{i=0}^{N-1} \alpha_{L-1}(i) \tag{2}$$

The forward algorithm only requires about $L \times N^2$ multiplications. For the second problem, the backward algorithm is a solution:

$$\beta_l(i) = P(O_{l+1}, O_{l+2}, \ldots, O_{L-1}|x_l = s_i, \lambda) \tag{3}$$

$\alpha_l(i)$ and $\beta_l(i)$ can be computed recursively. For all observations and states, define:

$$\gamma_l(i) = P(x_l = s_i|O, \lambda) \tag{4}$$

Since $\alpha_l(i)$ measures the relevant probability up to time l and $\beta_l(i)$ measures the relevant probability after time l.

$$\gamma_l(i) = \frac{\alpha_l(i)\beta_l(i)}{P(O|\lambda)} \tag{5}$$

From the definition of $\gamma_l(i)$ it follows that the most likely state at time l is the state s_i for which $\gamma_l(i)$ is maximum. The sizes of the matrices are fixed in the last problem, but the elements of matrices are to be determined.

$$\gamma_l(i, j) = P(x_l = s_i, x_{l+1} = s_j|O, \lambda) \tag{6}$$

$$\gamma_l(i, j) = \frac{\alpha_l(i)a_{ij}b(o_{l+1})\beta_{l+1}(j)}{P(O|\lambda)} \tag{7}$$

$$\gamma_l(i) = \sum_{j=0}^{N-1} \gamma_l(i, j) \tag{8}$$

Algorithm 1 Learning the First and Second Levels Matrices

$a_{ij} = P(state\ s_j\ at\ l+1 \mid state\ s_i\ at\ l)$

$b_j(k) = P(observation\ k\ at\ l \mid state\ s_j\ at\ l)$

For i=0 *to* N-1

  $\pi_i = \gamma_{0i}(i)$

For i=0 *to* N-1

  For j=0 *to* N-1

    $a_{ij} = \sum_{l=0}^{L-2} \gamma_l(i,j) \Big/ \sum_{l=0}^{L-2} \gamma_l(i)$

For j=0 *to* N-1

  For k=0 *to* M-1

    $b_j(k) = \sum_{\substack{l \in \{0,\ldots,L-2\} \\ o_l = k}} \gamma_l(j) \Big/ \sum_{l=0}^{L-2} \gamma_l(j)$

The solution for the last problem can be summarized in algorithm 2.

Algorithm 2 Estimating the Parameters of the Model

Initialize $\lambda(A,B,C,D,\pi)$.

Compute $\alpha_l(i)$, $\beta_l(i)$, $\gamma_l(i)$ *and* $\gamma_l(i,j)$ *using algorithm* 1.

Re-estimate the model $\lambda(A,B,C,D,\pi)$.

If $P(O|\lambda)$ *increases, go to* 2.

The algorithm 3 shows the general process of hierarchical hidden Markov model which is used to generate context-aware recommendations.

Algorithm 3 The Proposed Hierarchical Hidden Markov Model for CARS

Initialize $\lambda(A,B,C,D,\pi)$ *to small random values.*

For each user:

  For each user's feedback sequence:

    Re-estimate the model $\lambda(A,B,\pi)$ *based on given observation sequence.*

    Find the most likely first-level state sequence based on observation sequence.

    Re-estimate the model $\lambda(C,D,\pi)$ *based on first-level state sequence.*

    Find the most likely second-level state sequence based on first-level sequence.

    Predict next context using $\lambda(C,D,\pi)$.

    Predict next item using $\lambda(A,B,C,D,\pi)$.

For each user feedback sequence, matrices A and B are updated first. In other words, for each sequence, the first-level transition probabilities and the probability matrix of the observations (between the first level states and the items) are updated. In the next step, the sequence of states at first level with maximum likelihood is found. This sequence is equivalent to the sequence of transitions that most closely resembles user contextual states and shows the user feedback sequence. To find this sequence, the Viterbi algorithm is used [17]. Then the sequence found at the first level is used to update the matrices C and D. The procedure used at this step is the procedure used to update the matrices A and B. The only difference is the type of inputs, the parameters C and D are updated based on the sequences found in the first level, while the parameters A and B are re-estimated according to the sequence of the observed items.

In the next step, the state sequence is determined by the maximum likelihood for the second level. 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), the proposed model predicts the user's next context. Given the predicted context and the observations probability matrix B, the proposed model produces recommendations that match the predicted context.

After training the model, the multiplication of C and D matrices is used to predict the next context for the user. Then, by multiplying the matrices A and B, the probability of recommending each item is calculated based on the contexts. Finally, by using these probabilities and the predicted context, the top-N recommendations are predicted with the highest probability that correspond to predicated context. The overall process of hierarchical hidden Markov model for CARS can be seen in Fig. 2.

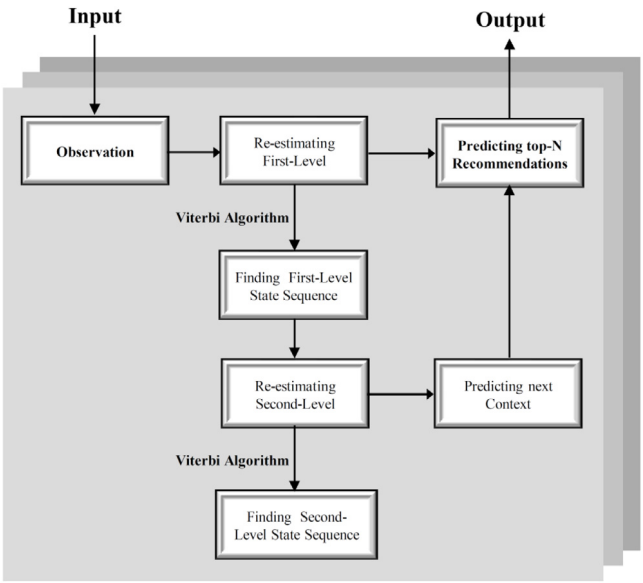

Fig. 2. Hierarchical hidden Markov model for CARS.

4. Experiments

This section describes a series of experiments to evaluate the proposed sequential recommendation approach against a number of other popular methods. This paper has used two popular data sets to evaluate the performance of the proposed model. The following sections describe the data sets and implementation results.

4.1. Data sets

There are several kinds of data sets for RS. This paper conducted a set of experiments with real usage data on the Last.fm and Netflix. Last.fm collects data on users' music listening activity. This data is used by Last.fm to make personalized recommendations at the online radio station. Celma collected apart of Last.fm data set that contains time-stamped records of users' music listening behavior [18]. It has 19,150,868 ratings, 992 users and 176,948 artists who were listened to.

Netflix data set contains 100,480,507 ratings, 17,770 movies and 480,189 users [19]. The data set consists of users' ratings on movies along with the time-stamp of rating. The proposed model estimates which movies the current user will rate in the next time period.

4.2. Baseline and state-of-the-art methods

The baseline approaches used in experiments are as followed:

HMM: Recommendation method based on the standard HMM model [16]. Similar to HHMM, the number of hidden states is set to 20 for training the HMM.

Sequential Pattern Mining: Recommendation method based on contiguous sequential patterns was used as one of the baselines in experiments [14]. The support value of sequential patterns was tuned and set to 0.01.

Item-based Markov Modeling: As another baseline sequential recommendation method, a kth-order Markov model was built where states corresponded to all item subsequences of length k observed in the training data set, and actions corresponded to different items. For each user, the last k items were used to generate recommendations. The transition probability for state s_i and action a_j was computed based on the training data set 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 approach was used as one of the baseline methods. The cosine metric was used to calculate the similarity of users and the size of the neighborhood for each user was set to 100.

Bayesian personalized ranking Matrix factorization (BPRMF): Matrix factorization using Bayesian personalized ranking was used as another baseline method in experiments [20]. This method was trained for 30 factors.

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

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

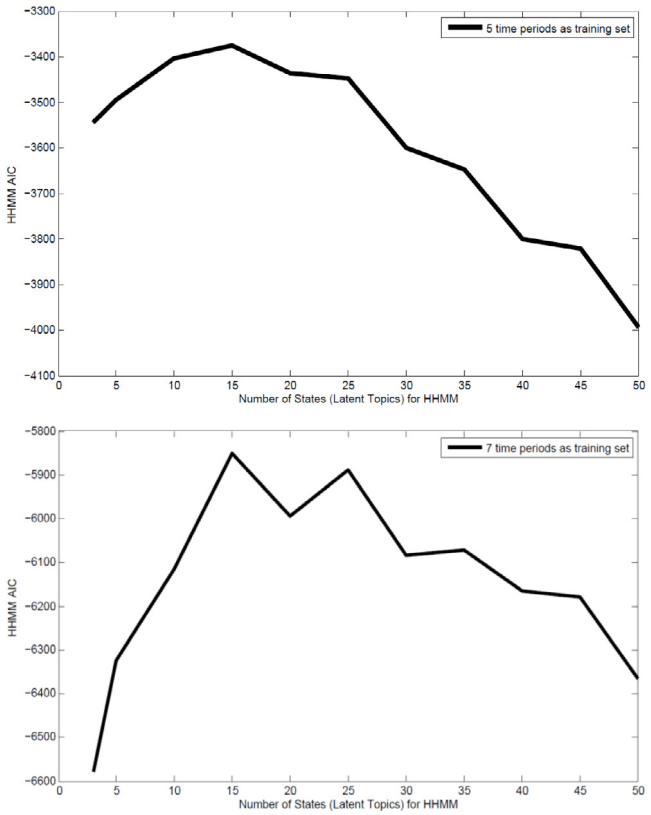

Fig. 3. AIC criterion for hierarchical hidden Markov model.

4.3. Model selection

Akaike information criterion (AIC) is a criterion to select the number of latent states that is closest to the true model given observation [21]. This paper used AIC criterion to determine the optimal number of latent states for HMM and HHMM.

$$AIC(\lambda) = ML_{\lambda} - P_{\lambda} \tag{9}$$

Where λ indicates an HHMM with a certain number of latent states, ML_{λ} is the maximum likelihood of the model, and P_{λ} is the number of parameters in the model. AIC increases with likelihood but decreases the complex model. AIC prescribes that one should select the model with the highest AIC score.

The optimal number of latent states is determined using AIC criterion. 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 complexity of the model increases as a square of the number of states, but the complexity of the model grows only linearly with the size of the data set (see Fig. 3).

4.4. Performance measures

The performances of the proposed model and compared methods are evaluated by precision and recall. These two measures are among the common metrics used for online evaluation of recommendation algorithms where the binary type of feedback is available. The precision and recall are computed as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{10}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{11}$$

Where *TP* represents the number of correct recommendations, and *FP* indicate the number of incorrect recommendations. *FN* indicates the number of relevant items that are not included in the recommendation list. A recommendation is assumed to be correct only if it appears in the test data. Another commonly used measure is *F*-measure that is defined as [22]:

$$F\text{-}measure = \frac{2 \times Precision \times \text{Recall}}{Precision + \text{Recall}} \tag{12}$$

Table 2
Comparing results from the proposed model against the results listed in [16] on Netflix data set.

Methods	Top 5			Top10		
	Precision	Recall	F-measure	Precision	Recall	F-measure
HHMM	0.1128	0.0335	0.0517	0.1385	0.0532	0.0769
Dynamic Model	0.1621	0.0310	0.0503	0.1452	0.0529	0.0748
Aspect Model	0.0652	0.0052	0.0093	0.0636	0.0100	0.0166
User-User Similarity	0.0798	0.0067	0.0121	0.0762	0.0125	0.0210
Popular	0.0770	0.0061	0.0110	0.0731	0.0111	0.0189
Link Analysis	0.0780	0.0062	0.0113	0.0742	0.0114	0.0194
Katz-CWT	0.1563	0.0281	0.0458	0.1397	0.0465	0.0665

Table 3
The results for 978 users and 7151 artists (artists who have been listened by at least 20 users) include on Last.fm data set.

Methods	Top 5			Top10		
	Precision	Recall	F-measure	Precision	Recall	F-measure
HHMM	0.0328	0.0132	0.0188	0.0248	0.0230	0.0239
Dynamic Model	0.0389	0.0135	0.0200	0.0340	0.0245	0.0285
Aspect Model	0.0282	0.0080	0.0122	0.0259	0.0152	0.0186
User-User Similarity	0.0409	0.0137	0.0201	0.0361	0.0240	0.0283
Popular	0.0292	0.0097	0.0140	0.0262	0.0168	0.0199
Link Analysis	0.0374	0.0124	0.0182	0.0329	0.0209	0.0250
Katz-CWT	0.0371	0.0134	0.0189	0.0329	0.0229	0.0263

Table 4
The results for 924 users and 1342 artists (artists who have been listened by at least 100 users) include on Last.fm data set.

Methods	Top 5			Top10		
	Precision	Recall	F-measure	Precision	Recall	F-measure
HHMM	0.0413	0.0312	0.0355	0.0425	0.0526	0.0470
Dynamic Model	0.0473	0.0314	0.0369	0.0423	0.0525	0.0469
Aspect Model	0.0308	0.0172	0.0217	0.0282	0.0310	0.0290
User-User Similarity	0.0482	0.0297	0.0357	0.0434	0.0526	0.0470
Popular	0.0326	0.0190	0.0234	0.0295	0.0326	0.0303
Link Analysis	0.0424	0.0247	0.0307	0.0374	0.0436	0.0397
Katz-CWT	0.0437	0.0289	0.0337	0.0393	0.0501	0.0430

The top 5 and top 10 highest scoring items are recommended for the user.

Popularity Bias was used in experiments to compare the diversity of different recommendation approaches. This paper followed the same approach as in [23] to calculate this measure. 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 method, the top $N = 10$ recommendations generated for each test case were then analyzed to compute the normalized distribution of each category.

4.5. Experimental results

This section focuses on comparing the performance of HHMM against other methods. The tested models include the proposed HHMM and the number of other approaches listed in [16]. HHMM needs sequences of adequate length to learn the transition probabilities. The collected data over each time period is used to train the algorithm. All tested models were implemented on MATLAB R2015. All experiments were conducted on a Microsoft Windows 7 with two 2.67 GHz dual-core CPUs and 8 GB RAM.

In the first set of experiments, users who have rated at least 2000 movies on Netflix data set are selected. This results in a data set with 2,626,320 ratings, 5265 movies, and 1212 users. The proposed model and the compared approaches are applied to this data set. The number of states for HHMM, dynamic model and Aspect model was selected using AIC criterion. The precision and recall for Netflix data set at the top 5 and top 10 levels are shown in Table 2.

We can see an advantage of the HHMM over the other methods. The best among the methods compared to the proposed HHMM is a dynamic model. Also, the proposed model was evaluated on Last.fm data set on the task of predicting the artists a user will listen to in the past. This data set contained the time-stamped sequence of artists that each user had listened to during the specified time period. The precision and recall scores, on last.fm data set for artist recommendation, are computed in two sparsity levels of data. The length of the time period was set to one month. The precision and recall at top 5 and top 10 levels are shown in Tables 3 and 4.

Other experiments are done based on users' music listening activities gathered from Last.fm website between 2009/01/01 to 2009/05/31. The first four months of the data is used for training the methods and the last month for evaluation of

Table 5
The performance evaluation of HHMM and baseline approaches.

Methods	Top 5			Top10		
	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

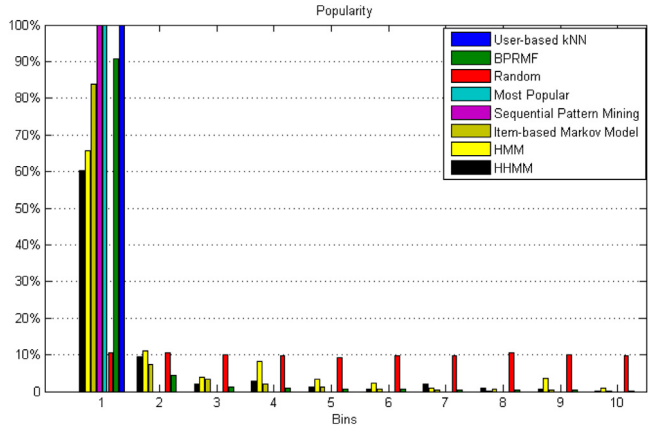

Fig. 4. Comparison of popularity bias of HHMM and baseline methods.

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 51,759. The test data contained 462 users. Given a user, each of the competing methods generated a set of N recommendations which were then evaluated based on the test data.

This part focuses on comparing the performance of HHMM against baseline approaches that are introduced in Section 3. Table 5 shows the recall and precision at top 5 and top 10 levels for each of the competing methods. According to this table, HMM has slightly better precision in comparison to HHMM and also has a significantly higher precision compared to the rest of the baseline methods. However, HHMM has the highest recall and achieves the best overall F-measure.

Fig. 4 compares the popularity bias of HHMM and other methods. As expected, the random method 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, and sequential pattern mining have the highest popularity bias. All the recommendations produced by these three methods 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

Recommender systems often generate recommendations based on user's prior preferences. Many model-based recommender systems, such as matrix factorization or neighborhood-based methods, do not consider changes in user's interests to recommend items [24]. Context-aware recommender systems take into account changes in user preferences by modeling them in time. Using users' context is challenging because it is not always possible to obtain all the contextual information. Also, adding various types of contexts to recommender systems increases its dimensionality and sparsity. This paper proposed a novel recommendation model based on hierarchical hidden Markov models to adapt to users' preference change as a result of contextual changes. The proposed model finds the maximum likelihood of transitions between model states for each user and computes the probability distribution for predicting the user's next context. Finally, it uses the modeled context to predict personalized recommendations. The experimental results show that using the predicted context can improve the diversity of the recommendations.

Acknowledgment

This work was supported by the University of Bonab, Iran.

References

[1] H. Shin, S. Kim, J. Shin, X. Xiao, Privacy enhanced matrix factorization for recommendation with local differential privacy, IEEE Trans. Knowl. Data Eng. 17 (6) (2018) 734–749.
[2] G.D. Abowd, A.K. Dey, P.J. Brown, N. Davies, M. Smith, P. Steggles, Towards a better understanding of context and context-awareness, in: International Symposium on Handheld and Ubiquitous Computing, Springer, 2001, pp. 304–307.
[3] G. Adomavicius, B. Mobasher, F. Ricci, A. Tuzhilin, Context-aware recommender systems, AI Mag. 32 (3) (2011) 67–80.
[4] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions, IEEE Trans. Knowl. Data Eng. 17 (6) (2005) 734–749.
[5] G. Adomavicius, A. Tuzhilin, Context-Aware Recommender Systems, in: Recommender Systems Handbook, Springer, 2011, pp. 217–253.
[6] S. Ebrahimi, N.M. Villegas, H.A. Müller, A. Thomo, Smarter deals: a context-aware deal recommendation system based on the smarter context engine, in: Proc. 2012 Conf. of the Center for Advanced Studies on Collaborative Re- search, IBM Corp, 2012, pp. 116–130.
[7] J. Lu, D. Wu, M. Mao, W. Wang, G. Zhang, Recommender system application developments: a survey, Decis. Support Syst. 74 (2015) 12–32.
[8] H. Ma, T.C. Zhou, M.R. Lyu, I. King, Improving recommender systems by incorporating social contextual information, ACM Trans. Inf. Syst. 29 (2) (2011).
[9] U. Panniello, M. Gorgoglione, Incorporating context into recommender systems: an empirical comparison of context-based approaches, Electron. Commer. Res. 12 (1) (2012) 1–30.
[10] N.M. Villegas, Context management and self-adaptivity for situation-aware smart software systems, Ph.D. thesis, University of Victoria, 2013.
[11] M.H. Aghdam, N. Hariri, B. Mobasher, R. Borke, Adapting recommendations to contextual changes using hierarchical hidden Markov models, in: Proc. of the 9th ACM Conf. on Recommender Systems, 2015, pp. 241–244.
[12] H. Lieberman, T. Selker, Out of context: Computer systems that adapt to, and learn from context, IBM Syst. J. 39 (3.4) (2000) 617–632.
[13] W. Woerndl, J. Schlichter, Introducing context into recommender systems, in: Proc. of the AAAI workshop on Recommender Syst. in E-commerce, 2007, pp. 138–140.
[14] N. Hariri, B. Mobasher, R. Burke, Context-Aware music recommendation based on latent topic sequential patterns, in: Proc. of the sixth ACM conference on Recommender Systems, 2012, pp. 131–138.
[15] J. Pitkow, P. Pirolli, Mining longest repeating subsequences to predict world wide web surfing, in: Proc. USENIX Symp. On Internet Technologies and Syst., 1999, p. 1.
[16] N. Sahoo, P.V. Singh, T. Mukhopadhyay, A hidden Markov model for collaborative filtering, MIS Q. 36 (4) (2012) 1329–1356.
[17] M.S. Ryan, G.R. Nudd, The Viterbi Algorithm, 1993.
[18] O. Celma, Music Recommendation and Discovery -The Long Tail, Long Fail, and Long Play in the Digital Music Space, Springer, 2010, (Chapter 3).
[19] J. Bennet, S. Lanning, The Netflix Prize, in: KDD Cup Workshop, 2007, http://www.netflixprize.com.
[20] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, BPR: Bayesian personalized ranking from implicit feedback, in: Proc. of the Twenty-Fifth Conf. on Uncertainty in Artificial Intelligence, 2012.
[21] H. Akaike, A new look at the statistical model identification, IEEE Trans. Automat. Control 19 (6) (1974) 716–723.
[22] C.J. Van Rijsbergen, Information Retrieval, book.
[23] F.G.D. Jannach, L. Lerche, G. Bonnin, What recommenders recommend - an analysis of accuracy, popularity, and sales diversity effects, in: User Modeling, Adaptation, and Personalization, Springer, 2013, pp. 25–37.
[24] M.H. Aghdam, M. Analoui, P. Kabiri, Collaborative filtering using non-negative matrix factorisation, J. Inf. Sci. 43 (4) (2017) 567–579.