Context-Aware Music Recommendation Based on Latent Topic Sequential Patterns

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

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

Robin Burke
DePaul University
School of Computing
Chicago, IL 60604, USA
burke@cs.depaul.edu

ABSTRACT

Contextual factors can greatly influence the users' preferences in listening to music. Although it is hard to capture these factors directly, it is possible to see their effects on the sequence of songs liked by the user in his/her current interaction with the system. In this paper, we present a context-aware music recommender system which infers contextual information based on the most recent sequence of songs liked by the user. Our approach mines the top frequent tags for songs from social tagging Web sites and uses topic modeling to determine a set of latent topics for each song, representing different contexts. Using a database of human-compiled playlists, each playlist is mapped into a sequence of topics and frequent sequential patterns are discovered among these topics. These patterns represent frequent sequences of transitions between the latent topics representing contexts. Given a sequence of songs in a user's current interaction, the discovered patterns are used to predict the next topic in the playlist. The predicted topics are then used to post-filter the initial ranking produced by a traditional recommendation algorithm. Our experimental evaluation suggests that our system can help produce better recommendations in comparison to a conventional recommender system based on collaborative or content-based filtering. Furthermore, the topic modeling approach proposed here is also useful in providing better insight into the underlying reasons for song selection and in applications such as playlist construction and context prediction.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering

Keywords

recommender systems, collaborative filtering, context-aware recommendation

1. INTRODUCTION

Traditional recommender systems have been extensively used in various applications to make recommendations based on users' history of preferences. However, in some applications, failure to consider the users' current situations may result in considerable performance degradation in recommendation effectiveness because users may have different preferences for items in different contexts. To address this problem, the notion of *context-awareness* has been the focus of many research projects.

Based on the classification presented in [1], knowledge of a recommender system about the contextual factors can be of three types: fully observable, partially observable, and unobservable. The contextual information is fully observable if all the relevant contextual factors, their structure and values are explicitly known. On the other hand, if only part of this knowledge is available then it is partially observable to the system. For example, if a restaurant recommender system knows that location, and time are the only important contextual factors and the values of these factors are explicitly given to the system, the contextual knowledge is fully observable in the system. But as it is more common, part of this information might be missing which makes the contextual information partially observable. In unobservable type of knowledge, no explicit information is available about the contextual factors. The way the recommender system will infer and represent context depends on the specific domain and the data available to the system.

In the domain of music recommendation, the setting in which our work is focused, context is usually not fully observable. Moreover, the contextual information may not be captured with a static set of factors, but rather, it is dynamic and should be inferred from users' interactions with the system. More specifically, context is reflected in the sequence of songs liked or played by the user in his/her current interaction with the system, such as a playlist. For example, in Pandora [1], users create different stations by selecting different track seeds or artists. The user can later play each of these stations based on his/her current preferences which can be influenced by different contextual factors such as mood, occasion, social setting, or the task at hand. Given a set of songs in which the user shows interest during an interaction, the recommender system should be able to recommend suitable songs for the current contextual state of the user.

In this paper, we present a music recommender system that captures the changing contextual states of the user

[1]http://www.pandora.com

based on the sequence of songs belonging to a playlist or an active interaction session with the system. The recommender system tracks changes in users' preferences and dynamically adapts to these changes. We use a topic modeling approach to map user's interaction sequence to a sequence of latent topics which capture more general trends in user's interests. The latent topics are generated from the top most frequent tags associated with songs, obtained from social tagging Web sites such as last.fm. In order to capture changes in the contextual states over time, we employ sequential pattern mining. Using a training set consisting of human-compiled playlists, sequential patterns are mined over the set of latent topics where each pattern represents a frequent sequence of transitions between topics representing contexts. Given a user's current interaction as the sequence of last w songs, the discovered patterns are used to predict the context for the next song. Additional interaction or ratings by the user during the same session may result in changes to the predicted context. The predicted context is then used to post-filter and re-rank the recommendations produced based on the whole history of the user's preferences.

Mining sequential patterns of topics instead of songs is useful in capturing the general characteristics of songs that are interesting for the user in a given context. Looking at users' interests at a more abstract level makes it easier to track and detect any changes in the users' preferences. Moreover, having topic-based instead of song-based patterns can be specifically useful in handling the cold start problem. A new song which hasn't occurred in the training data may not match sequential patterns obtained from song sequences, but it is likely to match topic-based patterns. Also, it helps to discover patterns at a higher confidence level which is particularly important when the training data for pattern mining is not large enough or too sparse.

In addition to song recommendation, our proposed approach can be used in various applications such as automatic playlist generation where the order of songs and the transition between them is meaningful and is one of the factors affecting the quality of playlists. For example, DJs have special techniques for continuous matching and ordering of songs in a mix. A possible usage scenario for our system would be to ask the user to select an initial sequence of songs for the playlist and produce recommendations based on matching patterns. The user adds one of the recommendations (or a new song outside the recommendation set) to the playlist, and the same process will be repeated again. As we will later discuss, if the user decides to suddenly change the music type in the playlist, our method is able to dynamically adapt to these changes. Similarly, our approach can be used for playlist recommendation, where the user inputs a playlist and looks for similar playlists. In other applications such as music radio, it is still important to track the order of songs liked by the user in order to determine any changes in his/her interests.

The remainder of this paper is structured as follows: Section 2 describes the details of our topic modeling module. In section 3, we introduce the sequential pattern mining component in our system. In section 4, the topic prediction algorithm is described and some evaluations for topic prediction are provided. Our context-aware music recommendation approach is described in section 5.

2. TOPIC MODELING FOR SONG REPRESENTATION

As previously discussed, our system infers the contextual information for each user based on the selection and order of songs in the user's current playlist. To track changes in song characteristics, each song is represented as a set of topics. Instead of manually defining a large number of features for each song, we use social tagging Web sites to automatically extract tags and generate topics.

In our system, the set of top tags for each song are retrieved from last.fm and those with frequency above a minimum threshold are selected. These tags describe various features of the songs including genre, artist name and the era, but they also describe users' attitudes toward the songs, including such feelings as sad, nostalgic, upbeat, and calm. Although people may have different and even contradictory opinions about some songs (particularly those that are related to "mood"), top tags with frequency above a minimum threshold capture the social opinion about each song. While some content-based audio features are not usually contained in tagging data, other characteristics such as cultural norms, references to some events, mood, and theme of the music can be effectively captured. These features can often be very helpful in explaining the commonalities in a set of songs selected by the user. For example, a user who selects "Beauty and the Beast", "The Little Mermaid - Part Of Your World" and "Aladdin - A Whole New World" as seed songs for a station is most likely interested in Disney movie soundtracks. While these songs can be totally different in structural features extracted from music audio, they have been assigned common tags such as "disney", "childhood", "soundtrack".

Instead of using individual tags as features, our system uses Latent topics to represent a contextual state. We use Latent Drichlet Allocation[4] topic modeling approach to reduce the dimensionality and the noise of the feature space, and also to capture the latent relationships between tags and songs. In order to do this, songs are taken as documents and tags as words. After fitting the topic model for K topics, the probability distribution over topics can be inferred for any given song. For each song in our database the set of *dominant topics* can be determined by selecting all topics with probabilities higher than a specific threshold value. This way, for each song a set of one or more topics is selected allowing a mapping of a song sequence into a sequence of latent topics. As we will explain in section 3 this process simplifies tracking the changes in song characteristics and user interests, and it will enable the prediction of future topics.

2.1 Relationships between Music Features and Topics

In this section we illustrate the characteristics of the playlist data set used in our experiments, and explore the potential relationships between some of the songs features and the selected dominant latent topics. Given a song title and the artist name, the set of top tags with frequency of at least 4 were retrieved from last.fm for all the songs in our database. Songs having a minimum of 5 tags were then selected to be used for building the LDA model. We fit a 30-topic LDA model to the set of 48862 selected songs.

Table 1 shows a set of most frequent tags for a sample of ten topics in the fitted model. Based on the frequent tags,

Table 1: Top Most Frequent Tags for a Sample of Topics

Topic#1	Topic#2	Topic#3	Topic#4	Topic#5	Topic#6	Topic#7	Topic#8	Topic#9	Topic#10	Topic#11	Topic#12
ambient	latin	death	60s	chill	beautiful	electronic	country	reggae	hop	industrial	hard
instrumental	world	thrash	oldies	downtempo	sad	electronica	americana	roots	hip	dark	stoner
soundtrack	streamable	black	roll	chillout	mellow	house	alt-country	rnb	rap	gothic	heavy
classical	spanish	heavy	50s	christmas	melancholy	techno	american	dub	hip-hop	goth	metal
beautiful	para	doom	rockabilly	lounge	acoustic	tranc	e french	slow	underground	electronic	nu-metal
age	bossa	brutal	top	electronic	chill	electro	singer-songwriter	ska	old	darkwave	progressive
chillout	fusion	melodic	500	trip-hop	soft	bass	texas	jams	coast	electro	american
experimental	musica	california	radio	electronica	slow	drum	acoustic	chill	hiphop	ebm	angry
movie	que	power	rolling	trip	melancholic	ambient	alt	dancehall	real	wave	crossover
atmospheric	nova	progressive	1960s	ambient	favourite	beat	chanson	jamaica	school	loved	lyrics
world	brazilian	gods	rhythm	hop	chillout	idm	old	jam	east	electronica	00s
ethereal	african	seixas	time	easy	ballad	experimental	southern	jackson	stream	live	hell
chill	party	speed	elvis	cool	singer-songwriter	club	roots	mellow	west	experimental	cool
calm	brasil	swedish	soundtrack	sexy	life	minimal	bluegrass	r&b	beats	post-punk	thrash
electronic	espanol	old	american	radio	easy	party	cash	smooth	york	synth	neo metal

these topics seem to be a mixture of different attributes of songs such as genre and era as well as other features such as mood and theme which represent the social feeling of the music. For example, frequent terms in topic#10 are related to hip-hop music and are representative of some of hip-hop subgenres such as *New York East Coast hip hop*, *West Coast hip hop*, *old school(skool) hip hop*, and *Gangsta rap*. Top terms in Topic # 7 are mostly describing techno/trance music; other terms such as *club* and *Party* describe the social context related to this topic. Top terms of topic#6 are descriptive of songs having a soft mood. Topic#4 is mostly describing rock music in the 50's and 60's.
The genre feature in our database consists of 115 classes. We map these classes to Yahoo! Music Genre Hierarchy which contains 19 categories at the first level. To visualize the relation of genre and LDA topics, each of the genre classes were represented as a vector of topic co-occurrences where the j^{th} entry in the vector of class i is the number of times topic j has been selected as dominant for a song of genre i. The co-occurrence vectors for genre classes have been created based on the set of 9743 songs in our database with known genre categories. The genre vectors were then normalized and projected into the two-dimensional space using Interactive Document Map (IDMAP) method while cosine-similarity was used for distance estimation.

The results of this visualization is shown in Figure 1. Nodes having the same ancestor at the first level are given the same shading or color. Some of the first order categories are marked in the figure. This visualization illustrates that in most cases, different subgenre classes with the same parent at the first level of the hierarchy are placed close to each other and in clusters.

A similar experiment was repeated for different eras of music between 1950 and 2000. The results shown in Figure 2 shows that subcategories of the same era are placed in clusters. Also, it is interesting to see that, the relative distance of different eras in the two-dimensional space is proportional to their time-based distances. For example, subclasses for 50's and 60's music are located close to each other. The same goes for subclasses of 90's and 80's music.

3. SEQUENTIAL PATTERN MINING

The underlying assumption behind our context-aware music recommendation is that a users' current context is reflected in the selected sequence of songs where each song is represented with a set of dominant topics. Based on our training database of playlists, our goal is to find the

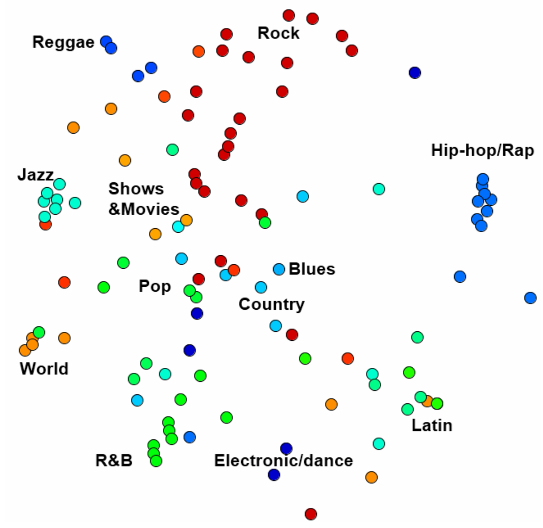

Figure 1: Music Genre Visualization by Topic-based Co-occurrence Analysis

topic-based sequential patterns that occur most frequently. Each frequent pattern is representative of a different context where the user has selected and ordered songs with specific characteristics captured in the topic sets.

Let D be a sequence database where each sequence corresponds to a playlist. In other words, each playlist is a sequence of songs where each song is represented by a set of topics. Let $A = \{t_1, t_2, ..., t_n\}$ be the set of all items which in our problem are LDA topics. A sequence $S =< x_1, x_2, ..., x_n >$ is an ordered list of elements x_i where each element is a subset of A. An item can occur at most once in each element but can occur multiple times in different elements of a sequence. Sequence $X =< x_1, x_2, ..., x_n >$ is a subsequence of $Y =< y_1, y_2, ..., y_m >$ and Y is a supersequence of X (or contains X) if there exists integers $1 < i_1 < i_2 < ... < i_n < m$ such that $x_1 \subset y_{i_1}$ and $x_2 \subset y_{i_2}$, ... , $x_n \subset y_{i_n}$. Support of X is defined as the number of supersequences of X in the database and is shown as $support(X)$. For a given threshold value $minSupport$, X is called a sequential pattern (SP), if $support(X) >= minSupport$. Contiguous sequential pattern (CSP) are a more restrictive form of sequential patterns which requires each pair of elements x_i and x_{i+1} to appear consecutively in a sequence Y which supports the pattern.

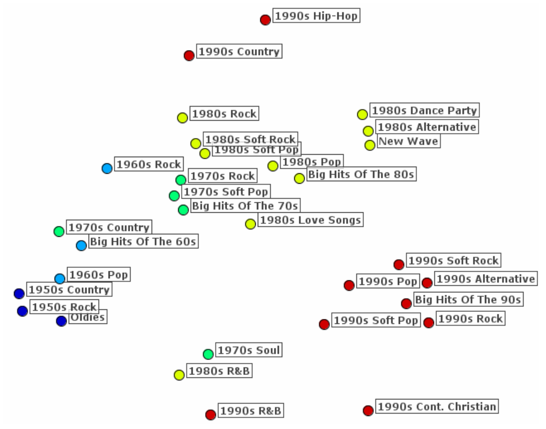

Figure 2: Music Era Visualization by Topic-based Co-occurrence Analysis

Various algorithms have been suggested for efficient sequential pattern mining. GSP [14] is one well-known example that works based on the Apriori principle which states that if a sequence is a sequential pattern then all of its subsequences must also be sequential patterns. Sequential patterns are mined using a candidate generation and pruning approach. Given a support threshold, at the start of the algorithm all frequent items are found. Each item represents a sequential pattern of size one. At each following step of the algorithm, the sequential patterns generated at the previous step will be used to generate new candidate sequences that respect the Apriori principle. The support values are found for each of the candidate sequences and the infrequent sequences are pruned. The remaining sequences are given to the next step and this procedure will continue until no sequential pattern is found in a step or no candidate sequence is generated. In spite of using Apriori pruning, methods similar to GSP are still generating a large number of candidates. Also, they require multiple passes on the database. Inspired with the idea of the FP-growth[6] approach, PrefixSpan[11] can be used for sequential pattern mining without generating candidate sequences at each step. In our experiments, we decided to use PrefixSpan as it has shown to efficient in mining patterns.

4. TOPIC PREDICTION USING SEQUENTIAL PATTERNS

The topic prediction module in our system takes a collection of topic sequential patterns as input and predicts a set of topics by matching the user's active session against the discovered patterns. We define the active session of user u, denoted by $h_u = < s_1, s_2, ..., s_n >$, as the sequence of last n songs that the user has shown interest in. Based on our preliminary experiments, to improve the average recall of topic predictions and to obtain a better balance between recall and precision, we assume that the selected dominant topics for each song are independent. Therefore, the user's activity sequence can be mapped to the corresponding set of single topic subsequences. Table 2 depicts an example user history h_u and the corresponding set of topic sequences h_u^i.

In the next step, each subsequence h_u^i is compared with the set of sequential patterns. A pattern, p, is accepted as a match for topic prediction, if its length is equal to $n+1$, and

Table 2: An example active session and the equivalent set of sequences

$h_u = << t_1, t_2 >< t_3 >< t_5, t_6 >>$
$h_u^1 = << t_1 >, < t_3 >, < t_5 >>$
$h_u^2 = << t_2 >, < t_3 >, < t_5 >>$
$h_u^3 = << t_1 >, < t_3 >, < t_6 >>$
$h_u^4 = << t_2 >, < t_3 >, < t_6 >>$

the prefix of size n of p, shown as $prefix_n(p)$, is contained in h_u^i. The set of topics contained in the $(n+1)^{th}$ element of p are selected as candidate topics and the recommendation confidence is calculated as the ratio of support of p to support of $prefix_n(p)$. The recommendation module suggests those topics that have a confidence above a threshold, min_c. Note that if different patterns select a topic with different confidence values, then the maximum of those confidences is selected as the prediction score.

In many situations, a user's preferences in music may change during an active session. If the active session is small, it can result in low precision in predictions because some of the longer patterns that better capture user's behavior are not being used. On the other hand, if the window is large, then the chance that the user's preference changes in this window is high, making it less likely to find a matching sequential pattern. That leads to low recall in predictions made by the system.

In order to overcome this problem, we use the all-k^{th}-order method proposed in [12] for Markov chain models. This idea has been extended in [10] to the context of general sequential patterns. To follow this approach, first, the recommendation engine uses the current active session as an input. If the engine cannot generate any recommendations, the size of active session window is iteratively decreased by removing the oldest song until a recommendation is generated or the window size becomes 0.

To show how the all-k^{th}-order method can be useful in detecting the changes in the user's preferences, consider a playlist of length 9 shown in Table 3. For each song in the playlist, artist name, popular tags from last.fm, and the set of dominant topics are presented. In this example, the user starts by listening to mellow, acoustic rock. The first four songs have tags such as *rock*, *acoustic*, *guitar*, *mellow*, *singer-songwriter* and topic #6 is common between all these four songs. As shown in Table 1, some of the mentioned tags are appearing as the most frequent terms for this topic.

At the fifth song, there is a sudden change from *acoustic* music to *electronic* and *trip-hop*. Following the all-k^{th}-order method, a match is not found until the sixth iteration (that is when we are only considering the last four songs). The change from *acoustic* to *electronic* did not appear in any sequential pattern seen before, so the system was unable to match any patterns in the first five iterations. For the sequence containing the last four songs, the following patterns are selected:

$$\begin{array}{c}
23 \rightarrow 6 \rightarrow 30 \rightarrow 20 \rightarrow 6 \\
23 \rightarrow 6 \rightarrow 30 \rightarrow 6 \rightarrow 6 \\
23 \rightarrow 6 \rightarrow 18 \rightarrow 20 \rightarrow 6 \\
23 \rightarrow 6 \rightarrow 18 \rightarrow 6 \rightarrow 23 \\
23 \rightarrow 6 \rightarrow 18 \rightarrow 6 \rightarrow 6 \\
23 \rightarrow 25 \rightarrow 18 \rightarrow 6 \rightarrow 6 \\
23 \rightarrow 25 \rightarrow 18 \rightarrow 20 \rightarrow 23 \\
23 \rightarrow 25 \rightarrow 18 \rightarrow 6 \rightarrow 23
\end{array} \tag{1}$$

Table 3: Example of a playlist with corresponding dominant topics and popular tags

Time	Song Title	Artist Name	Popular Tags	Dominant Topics
1	why Georgia	John Mayer	singer-songwriter, mellow, relaxing, chill, male vocalist, easy listening, acoustic, 00's, guitar, rock, happy	6
2	bubble toes	Jack Johhnson	singer-songwriter, chill, acoustic, mellow, rock, summer, surf, male vocalists, pop, relaxing, guitar, happy	6
3	rose parade	Eliot Smith	singer-songwriter, indie rock, folk, acoustic, mellow, chillout, relaxing, bittersweet, lo-fi	6, 20, 23
4	may angels lead you in	Jimmy Eat World	alternative rock, ballads, calm, beautiful, nice, soundtrack, favorites	6, 28
5	sexy boy	Air	electronic, electronica,french, chillout, trip-hop, ambient, downtempo, sexy, 90s, alternative, easy listening, guitar, mellow, relax, female vocal	7, 5
6	what you are	Drill	soundtrack, 90s, alternative, atmospheric, female vocalists, indie, dreamy	23
7	flake	Jack Johhnson	singer-songwriter, acoustic, chill, alternative, rock, male vocalists, easy listening, driving	6, 25
8	what is life	Shaun Mullins	cover, beatles cover, rock, 90s, soundtrack, brass , pop rock, alternative rock, folk, brass	30, 18
9	all mixed up	red house painter	indie, rock, acoustic, 90's, cover, mellow, pop, folk, dreamy, singer-songwriter, sadcore, summery, sweet, alternative rock, female vocalist	6, 20

Based on these patterns, either topic 6 or 23 will have high probability to appear next.

4.1 Topic Prediction Evaluation

In order to evaluate the performance of our topic prediction approach, we compared our approach to several well-known pattern mining approaches. Furthermore, section 4.2 describes how Markov models can be used for topic prediction and compares the results with our sequential pattern mining method.

The database used for this experiment contains 28,963 user-contributed playlists from "Art of the Mix" website[2] in January 2003. This dataset consists of 218,261 distinct songs for 48,169 distinct artists. The average number of songs per playlist is 19.8 and the average number of artists in playlists is 17.1. Top tags were retrieved from the last.fm website for about 71,600 songs in our database. For the rest of the songs either a match was not found in last.fm or there were no top tags available for that song. For the experiment, we built the LDA model for 30 topics and set the threshold for selecting dominant topics to 0.25.

About 7,051 playlists with enough tags (at least 8) for at least 10 songs in the playlist, were used for evaluating the topic predictions. The selected playlists contain 21,783 unique songs. The algorithm was then evaluated using 10-fold cross validation. In each run of the algorithm, sequential patterns were generated using 9 folds of the data and the remaining fold was used as the test set. The support threshold for choosing a sequence as frequent pattern was set to 30.

For each playlist in the evaluation set, the last $w = 7$ songs were selected as the user's active session, the last song was removed and the dominant topics associated with that song were used as target set. Given a confidence threshold, α, the topic prediction module makes recommendations based on the remaining songs in the user's active session. The recommended topics all have recommendation scores of at least α. Figures 3 and 4, compare the precision and recall of the previously mentioned sequential pattern mining approaches for different levels of confidence. According to these figures, contiguous sequential patterns (CSP) produce more accurate results while achieving lower recall levels than general (open) sequential patterns (SP). Also, for $k = 7$, we tested the all-k^{th}-order contiguous sequential patterns, shown as all-k-th order CSP, and all-k^{th}-order general sequential patterns, shown as all-k^{th}-order SP. Similar to the results reported in [10], applying all-k^{th}-order approach reduces precision, but significantly improves recall for both SP and CSP methods. Based on precision and recall figures, the best F-score is achieved by all-k^{th}-order SP for confidence level between 0.3 to 0.4. Therefore, in our system we used the all-k-k^{th}-order sequential pattern mining method and accepted the topic predictions with confidence score of at least 0.3.

[2]http://www.artofthemix.org/

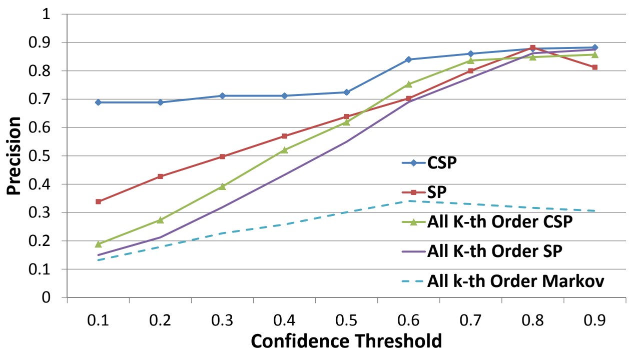

Figure 3: Topic Prediction Precision

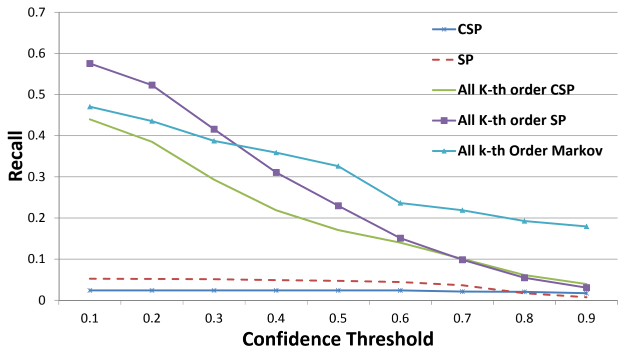

Figure 4: Topic Prediction Recall

4.2 Topic prediction using Markov Models

Given a training database of music-listening sessions, a k^{th}-order Markov model can be built where states corresponds to all song subsequences of length k observed in the training data, and actions corresponds to different songs. In this model, the last k actions are used to make predictions about the next action. The transition probability for state s_i and action a_j is computed based on the training sequence database and as the normalized frequency of times a_j has been observed to follow s_j.

For many problems, first-order Markov models have low accuracy in making predictions. Although higher-order modeling can improve the prediction precision, it dramatically increases the model complexity and adds to the running time of the algorithm. Moreover, the algorithm will achieve a much lower recall level in making predictions. Although using the all-k^{th}-order approach can improve coverage, it adds to the state-space complexity even more.

To compare the performance of the Markov model with the sequential pattern mining approaches, the same experiment as in the previous section was set up and the all-k^{th}-order Markov model was used for topic prediction. The order of the model, was similarly set to $k = 7$. The precision and recall of the predictions are compared with sequential pattern mining methods, and are depicted in Figures 3, 4. According to these figures, even without considering the space complexity and long run-time of the all-K^{th}-order Markov model, sequential pattern mining performs better in making a proper balance in recall and precision and achieving a higher F-score.

5. CONTEXT-AWARE MUSIC RECOMMENDATION

As previously discussed, given a window of last n songs listened by the user, the topic prediction component makes predictions about the dominant topics of the next song. This is given as the contextual information to the music recommender module.

As described in [2], there are different ways to incorporate the contextual information into the recommendation algorithm. In contextual pre-filtering approaches, the dataset is first filtered, the recommendations are then provided based on the contextualized dataset. On the other hand, a contextual post-filtering method generates recommendations similar to traditional recommender systems. It then filters and re-ranks the recommended items to generate contextual recommendations. In contextual modeling, context is added to the problem as an additional dimension.

In our system, we used a post-filtering approach to integrate the context information. For a given user, u, first an initial ranking is generated using a traditional recommendation algorithm. This initial ordering is generated based on the complete history of user's preferences. In our experiments we used user-based kNN algorithm and computed the similarities between users based on binary cosine similarity metric. Given a user u, the initial recommendation score for a candidate song s is calculated as in equation 2. In this formula, N_u represents the user's neighborhood and $m_{n,s}$ indicates whether neighbor n has listened to song s.

$$CFScore(s) = \sum_{n \in N_u} similarity(u, n) \cdot m_{n,s} \tag{2}$$

In the next step, the recommendations are re-ranked using the contextual information extracted according to the recent preferences of the user (i.e. the last n songs). In order to do that, a contextual score is computed for each song in the recommendation list which shows the suitability of that song for the current context of the user. One heuristic to find the contextual score is to take the average topic probabilities for the candidate song over the set of predicted topics for the next song. Formally, given the active user's session as h_u, the context score of a candidate song s is calculated as follows:

$$contextScore(h_u, s) = \frac{\sum_{t_i \in predictedTopics(h_u)} p(t_i|s)}{|predictedTopics(h_u)|} \tag{3}$$

There are different ways to re-rank the initial ordering of recommendations. We followed equation 4 to compute the final recommendation scores which has shown to work well on our dataset. These scores are used to produce the final ranking of the recommended songs to the user.

$$\begin{aligned} &predictionScore(s) = \\ &(contextScore(s) + \alpha_1) \cdot (CFScore(s) + \alpha_2) \end{aligned} \tag{4}$$

In this formula, α_1 and α_2 are both smoothing constants and are set to 0.1 in our experiments.

5.1 Evaluation of Recommendations

The goal of this experiment is to determine the performance of the system in making good recommendations for a user's music-listening session (or playlist). We performed a 10-fold leave-one-out cross validation experiment using the same dataset which was used for topic prediction evaluation. For each playlist sequence in the test set, the last song was selected as the target item and was removed from the sequence. The music recommendation module was evaluated with respect to whether it was able to recommend back the removed song. If the song was found by the recommender, its rank in the overall recommendation list is recorded as top recommendations are more valuable for the user. The results for the leave-one-out cross validation was evaluated by computing the *Hit Ratio*, which computes the probability that the removed song is recommended as part of the top N recommendations. Formally, let's denote the top N recommendations for a given playlist, p, as $R_N(p)$. If in this playlist the removed target song, s_p, is part of $R_N(p)$, then it is considered a *hit*. For any given rank N, the hit ratio for the recommendation algorithm is computed as: $h(N) = |p \in testset : s_p \in R_N(p)|/|testset|$.

Figure 5 depicts the hit ratio at different levels of N less than 300 for our algorithm and some other methods including user-based kNN, Matrix Factorization using Bayesian Personalized Ranking (BPRMF) [13], and content-based recommender. Our approach for content-based recommendation and analysis of the results for this algorithm will be discussed in section 5.2. According to Figure 5, the context-aware recommender achieves higher hit ratio at all levels of N>30 in comparison to user-based kNN. This improvement is more significant as the number of recommendations increases.

The BPRMF recommender was trained for m=30 factors. The results show that both user-kNN and the context-aware recommender achieve higher hit ratio when the number of recommendations is less than 300. The average precision improvement of our method over user-based kNN is presented in Figure 6. As can be seen, our approach has higher average precison than user-based kNN at almost all cut-off levels. Similar to hit ratio results, the improvement in precision is more significant at cut-off values greater than 30.

Similarly, the average precision improvement over BPRM is depicted in Figure 7. The results show that at high ranks,

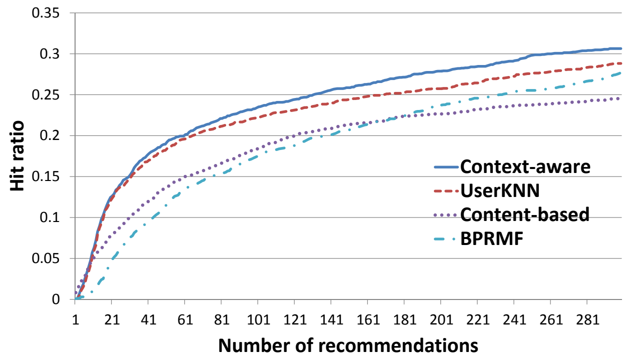

Figure 5: Hit ratio for different number of recommendations

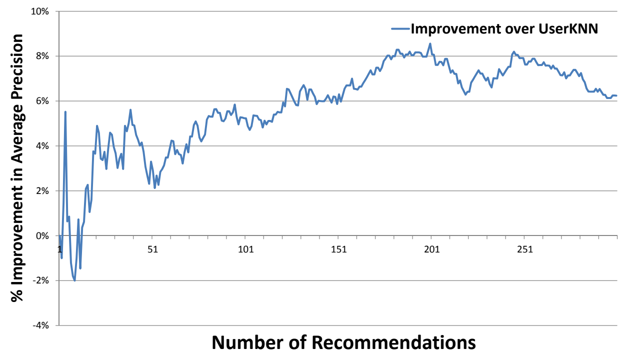

Figure 6: Average precision improvement of our method over user-based *k*NN

precision of the context-aware recommender can be more than four times larger than BPRMF algorithm. Based on the experiments in this section, our method achieves better performance than user-based *k*NN and BPRMF in terms of both hit ratio and precision.

5.2 Comparison with Content-based Recommender

This experiment compares the performance of our system against a content-based recommender which produces recommendations based on some of songs attributes including artist, genre, era, and album title. Each song is represented as a binary vector in the attribute space. Given a music listening-session for a user, p, in order to find the prediction score for a song s, first the k nearest neighbors of s are selected from the set of songs in p. The similarity of two songs is calculated as the binary-cosine similarity of their attribute vectors. The recommendation score for song s is then computed as the sum of similarities of s to its neighbors. The songs are then ranked based on the calculated recommendation scores.

As previously explained the dataset which was used in our experiments consists of 7,051 playlists with enough tags (at least 8) for at least 10 songs in the playlist and includes 21,783 unique songs. Beside artist attribute which is known for all the songs, the other three attributes might have missing values. About 9,250 songs have known genre attribute,

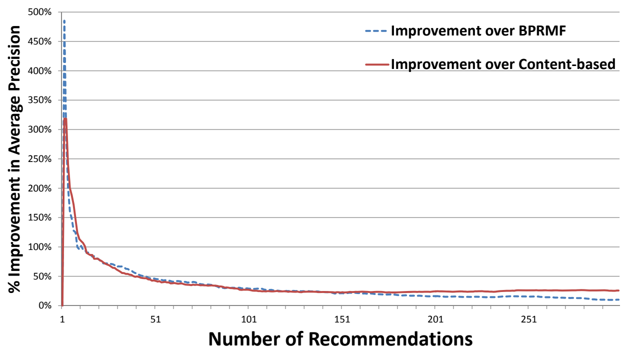

Figure 7: Average precision improvement of our method over BPRMF and content-based recommender

album is known for 11,380 songs, and era is known for 2,035 songs.

The content-based recommender was evaluated in a similar manner as described in the previous experiments. The hit ratio results of this experiment are shown in Figure 5. The hit ratio for the content-based recommender is much lower than the other methods. Also, according to the results in Figure 7, the precision of our approach can reach to more than three times that of the content-based algorithm. These results show that using song attributes alone will not result in optimal recommendation performance. In our future work, we are planning to include audio-content data and also investigate if a hybrid of context-aware method and these content-based approach can improve the performance.

6. RELATED WORK

Several researchers have previously investigated the use of contextual information in various applications of recommender systems. An interesting application of context-aware recommender systems is in mobile devices equipped with GPS or other sensors. For example, [7] uses temperature, weather, time and some other contextual factors received from sensors to suggest music to users. [8] uses a user's locations and selects music that fits a place of interest.

There has been some research related to context-aware playlist recommendation. In systems like stereomood [3], users can search for playlists that best fit their feelings or activities. For example, the user can search for happy playlists or playlists suitable for studying or partying. The users can assign mood and activity tags to each song which is played and this information is used by the system to group songs in relevant playlists. In other words, the mood-based recommendation engine uses the social mood given by active users to generate the playlists. In [9] a context-aware playlist recommendation system is proposed which provides users opportunity to browse their music by mood. This system uses a combination of audio and lyric content to classify music by mood. While mood is one of the important factors that can affect the playlist generation, it is not the only factor affecting the users' preferences.

The system described in [3] focuses on social context and proposes Poolcasting as a method to customize musical sequences for groups of listeners. Having a database of human-

[3]http://www.stereomood.com

compiled playlists, they analyzed co-occurrence of songs and artists to measure how much two songs or artists go well together in sequence. Co-occurrence analysis at song level requires much more training data in comparison to our method of finding patterns at topic-level. Also, using this approach, it is hard to find the association of the previously unseen songs/artists while our method is able to make predictions as long as tagging data is available for the songs in the playlist. Similar to our work, [5] defines context in music recommendation as the context of selecting and ordering the songs in the playlist. They generate playlists using both acoustic and social-network data. To extract the social network data, they analyzed a sample of Myspace artist network. For their evaluations, they proposed a method for comparing playlists. To compute the distance between playlists, topic modeling is used to represent songs as a mixture of topics based on their social tags. Each playlist with l songs is then represented as a $l \times d$ matrix where d is the number of topics in the mixture model. A form of cosine-similarity was used to find the similarity of two playlists. Although their representation of songs using topic modeling is similar to our work, our work is different from them in the sense that we are extracting topic-based sequential patterns and use them for context prediction.

7. CONCLUSIONS

This paper has presented a novel approach for context-aware music recommendation which infers the dynamic context of a user from the sequence of songs in his/her active interaction session with the system. Our system mines popular tags for songs from the last.fm Web site. The tag data is used by the topic modeling module which fits an LDA model and infers the topic probability distribution for songs. Each song is then represented as a set of topics which have probabilities above a certain threshold and each playlist is represented as a sequence of topicsets. The topic-based sequential patterns occurring frequently among playlists are then discovered based on a training dataset containing human-compiled playlists.

Given a user's interaction session, all matching topic-based sequential patterns are selected. The selected patterns are used to predict context of the next song. This information is used for contextual post-filtering of the recommendations given by a tradition recommendation algorithm such as the user-based kNN.

Our work differs from prior work in the area of pattern mining for music recommendation which discover patterns for song sequences or focus on analyzing co-occurrences of songs or artists. In contrast, our approach discovers patterns at a more abstract level rather than songs. This generalization makes it easier to track and detect any changes in the users' preferences. Also, it is useful in handling the cold start problem where a new song hasn't occurred in the training data.

The findings reported in this paper introduce numerous additional questions and areas of future work. We plan to extend our work to use other song features such as audio-content features in the recommendation algorithm and also investigate the relationship between LDA topics and audio features. Furthermore, we plan to follow a systematic approach for determining the best number of topics to be used in the topic modeling module.

8. REFERENCES

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

[2] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Recommender Systems Handbook*, pages 217–253. Springer, 2011.

[3] C. Baccigalupo. *Poolcasting: an intelligent technique to customise music programmes for their audience.* PhD thesis, Universitat Autonoma de Barcelona, 2009.

[4] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3, 2003.

[5] B. Fields. *Contextualize Your Listening:The Playlist as Recommendation Engine*. PhD thesis, University of London, 2011.

[6] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *ACM International Conference on Management of Data and Symposium on Principles of Database Systems, SIGMOD/PODS 2000, Dallas, TX, USA, May 15-18, 2000*, pages 1–12. ACM, 2000.

[7] H.Park, J. Yoo, and S. Cho. Context-aware music recommendation system using fuzzy bayesian networks with utility theory. In *Proceedings of the Third international conference on Fuzzy Systems and Knowledge Discovery*. Springer, 2006.

[8] M. Kaminskas and F. Ricci. Location-adapted music recommendation using tags. *User Modeling, Adaption and Personalization*, pages 183–194, 2011.

[9] O. Meyers. *A mood-based music classificationand exploration system.* Master's thesis, Massachusetts Institute of Technology, 2007.

[10] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Using sequential and non-sequential patterns for predictive web usage mining tasks. In *Proceedings of the IEEE International Conference on Data Mining (ICDM'2002)*, 2002.

[11] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. Prefixspan: Mining sequential patterns by prefix-projected growth. In *ICDE*, pages 215–224. IEEE Computer Society, 2001.

[12] J. E. Pitkow and P. Pirolli. Mining longest repeating subsequences to predict world wide web surfing. In *USENIX Symposium on Internet Technologies and Systems*, 1999.

[13] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI '09 Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence.*

[14] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *5th Intl. Conf. Extending Database Technology*, 1996.