Personalized Text-Based Music Retrieval

Negar Hariri, Bamshad Mobasher, Robin Burke
DePaul University, College of Computing and Digital Media
243 S. Wabash Ave, Chicago, IL 60604, USA
{nhariri, mobasher, rburke}@cs.depaul.edu

Abstract

We consider the problem of personalized text-based music retrieval where users' history of preferences are taken into account in addition to their issued textual queries. Current retrieval methods mostly rely on songs meta-data. This limits the query vocabulary. Moreover, it is very costly to gather this information in large collections of music. Alternatively, we use music annotations retrieved from social tagging Websites such as last.fm and use them as textual descriptions of songs.

Considering a user's profile and using preference patterns of music among all users, as in collaborative filtering approaches, can be useful in providing personalized and more satisfactory results. The main challenge is how to include both users' profiles and the songs meta-data in the retrieval model.

In this paper, we propose a hierarchical probabilistic model that takes into account the users' preference history as well as tag co-occurrences in songs. Our model is an extension of LDA where topics are formed as joint clusterings of songs and tags. These topics capture the tag associations and user preferences and correspond to different music tastes. Each user's profile is represented as a distribution over topics which shows the user's interests in different types of music. We will explain how our model can be used for contextual retrieval. Our experimental results show significant improvement in retrieval when user profiles are taken into account.

1 Introduction

As music libraries are increasing in size and popularity, various recommendation and retrieval methods have been proposed to facilitate accessing and browsing songs. Recommendation algorithms exploit previous user feedbacks to recommend songs that the user is likely to enjoy. Among these methods, content-based approaches use features extracted from music content or available meta-data to recommend similar songs to those previously liked by the user. On the other hand, collaborative filtering techniques utilize the profiles of all the users to find preference patterns of users and use them to make recommendations to a given user.

Unlike the recommendation methods, retrieval algorithms usually ignore the users’ history of preferences and retrieve songs that best match the issued queries. Existing text-based retrieval methods mostly rely on music meta-data and therefore limit the queries to a small vocabulary. Using a larger set of meta-data causes scalability problems for rapidly growing music databases as gathering this information for each of the songs requires music experts’ time and effort. As an alternative solution, social tagging data from Websites such as last.fm has been used in many application for representing songs features. In spite of the fact that the users assign different and sometimes contradicting tags to a given song, popular tags for a song are less prone to noise and are descriptive of different characteristics of the song such as genre, mood, theme, era, etc. Using these tags, the songs can be represented in the tag space and therefore, conventional retrieval approaches can be applicable for song retrieval.

Although recommendation and retrieval methods address different problems in general, in some cases the users’ profile (i.e. preference history) and their queries should be considered at the same time. Consider an application that helps users in building playlists. At each step of the process, the user can issue a query and is presented with a set of songs. Those songs accepted by the user are then added to the playlist. The retrieved songs at each step, should not only match the query but also be coherent with the other songs already in the playlist which represents the short-term preferences of the user. For example, if the previous songs are all rock and the user queries for a happy theme then it is most likely that the user is interested in songs which are both happy and rock.

In this paper, we introduce a statistical model for user’s profiles which can be used for personalized music retrieval. Our model is inspired by the Latent Dirichlet Allocation (LDA) model (Blei, Ng, and Jordan 2003), which was originally proposed for modeling text documents. LDA is a generative probabilistic model of a corpus. It models each document as a mixture of latent topics where each topic is a distribution over the vocabulary. Looking at each song as a document and its tags as the document words, the LDA model can be used to model a collection of songs. However, as the users’ profiles are not part of the model, it cannot be used for personalized retrieval.

Using our model on the other hand, each user is modeled as a mixture of topics where the topic proportions are inferred according to the set of songs previously liked by the user and their associated top tags. These topics capture the
tag associations as well as the user preference patterns and correspond to different music tastes.

Our model defines a generative process for a user’s profile (or similarly playlists) which generates songs and tags from a common set of topics. Each joint topic has a distribution over the set of songs and also a distribution over the tag vocabulary. As we will explain, having a common set of topics for tags and songs enables us to provide personalized retrieval.

2 Latent Dirichlet Allocation

As our model is an extension of LDA, we will first review this model. The plate diagram of the smoothed LDA model is shown in Figure 1. In the standard form of graphical models, nodes indicate random variables. Shaded nodes are observed random variables and unshaded nodes are latent random variables. A plate is shown as a box around a random variable and indicates replication. The edges between the nodes represent dependency between the variables.

In Figure 1, the outer plate corresponds to the documents in the corpus and is replicated N times, where N is the number of documents. The inner plate corresponds to words in each document and is replicated as many as the number of words in the document (M times). The LDA model can be viewed in terms of a generative process for documents. The complete notation used in Figure 1 is as follows:

- $\theta_i$ represents the topic distribution for document $i$.
- $z_{ij}$ denotes the topic for the $j^{th}$ word in document $i$.
- $\phi_k$ represents the word distribution for topic $k$.
- $w_{ij}$ denotes the $j^{th}$ word in document $i$.

In this model, $\theta$ is a Dirichlet random variable with parameter $\alpha$:

$$ P(\theta|\alpha) = \frac{\Gamma \left( \sum_{i=1}^{K} \alpha_i \right)}{\prod_{i=1}^{K} \Gamma (\alpha_i)} \theta_1^{\alpha_1-1} \ldots \theta_K^{\alpha_K-1} $$

and $\theta_i \geq 0, \sum_{i=1}^{K} \theta_i = 1$  \hspace{1cm} (1)

Where $\alpha$ is a k-dimensional vector with elements $\alpha_i > 0$ and $\Gamma(x)$ is the gamma function.

The variable $\phi$ is a $K \cdot V$ random matrix where $K$ is the number of topics and $V$ is the vocabulary size. Each row of this matrix is independently drawn from an exchangeable Dirichlet distribution with parameter $\beta$ and denotes the word distribution of a topic.

The document generation is based on the idea that each document is a mixture of topics, where a topic is a probability distribution over words.

To generate a new document $d$, first the distribution over topics shown as $\theta_{(d)}$ should be specified. For each word in the document a topic $z$ is selected based on $\theta_{(d)}$. According to $\phi_z$, a word is picked and is added to the document.

Given the LDA model with $K$ topics, the probability of a word $w$ given a document $d$ can be computed as in Equation 2. In this formula, $p(\theta|d)$ represents the inferred probability of $\theta$ given the observed words in the document $d$.

$$ p(w|d) = \sum_{z=1}^{K} \left( p(w|z) \cdot \int_{\theta} p(z|\theta)p(\theta|d) d\theta \right) \hspace{1cm} (2) $$

LDA has been successfully used for various applications such as query expansion, tag recommendation, document retrieval (Wang 2003). A different formulation of the LDA model has been used as a collaborative filtering approach for item recommendation (Blei, Ng, and Jordan 2003). For this application, users correspond to documents and words correspond to items. For a given user, the recommender ranks the items based on the probability of item given a user’s profile which can be computed similar to $p(w|d)$ in Equation 2.

3 Personalized Retrieval Model

This section introduces our probabilistic model for personalized text-based retrieval. Our approach is an extension of the LDA model. Similar to LDA, it captures the tag associations based on their co-occurrences in the same song. Furthermore, it also captures the songs associations based on their co-occurrences in the same user’s profile (or playlist).

The graphical representation of our model, which we call the personalized retrieval model, is shown in Figure 2. In this model, the outermost plate corresponds to users (or playlists) which is replicated $N$ times where $N$ is the number of users. Each user is modeled as a multinomial distribution over a set of topics where each topic has a distribution over the set of songs and tags. The random variables in this model are as follows:

- $\theta_i$ represents the topic distribution for user $i$.
- $z_{ij}$ denotes the topic for the $j^{th}$ song and its tags in $i^{th}$ user profile.
- $\mu_k$ represents the song distribution for topic $k$.
- $\phi_k$ represents the tag distribution for topic $k$.
- $s_{ij}$ denotes the $j^{th}$ song in the profile of user $i$.
- $t_{ijk}$ denotes the $k^{th}$ tag for the $j^{th}$ song in the $i^{th}$ user profile.

Similar to LDA, we assume that $\theta$ is a $k$-dimensional Dirichlet random variable with parameter $\alpha$, where $k$ is the number of topics.

The variable $\mu$ is a $K \cdot U$ random matrix where $K$ is the number of topics and $U$ is the number of unique songs. Each
row of this matrix is independently drawn from an exchangeable Dirichlet distribution with parameter \( \beta \) and denotes the song distribution of a topic. Similarly, the variable \( \phi \) is a \( K \times X \) random matrix where \( X \) is the size of tag vocabulary. Each row of this matrix is independently drawn from an exchangeable Dirichlet distribution with parameter \( \gamma \) and denotes the tag distribution of a topic.

A song \( s \) with \( w \) tags is assumed to be generated by first choosing a value for \( z \) and then sampling the song according to \( \mu \) and conditioning on the chosen value for \( z \), and also sampling \( w \) tags according to \( \phi \) and conditioning on the chosen value of \( z \). Therefore, it is assumed that each song and its tags are generated from the same topic.

More formally, the process for generating a user’s profile is as follows:

1. Choose \( \theta_\mu \sim \text{Dir}(\alpha) \).
2. Choose \( \phi_k \sim \text{Dir}(\gamma) \), for each topic \( k \).
3. Choose \( \mu_k \sim \text{Dir}(\beta) \), for each topic \( k \).
4. For each of the \( M \) songs, \( s_i \), in the user’s profile:
   (a) Choose \( z_i \sim \text{Multinomial}(\theta_\mu) \)
   (b) Choose \( s_i \sim \text{Multinomial}(\mu_k) \)
   (c) For each of the \( W \) tags, \( t_j \), associated with \( s_i \)
      i. Choose \( t_j \sim \text{Multinomial}(\phi_{z_i}) \)

Given \( \mu \) and \( \phi \), the joint distribution of topic mixtures \( \theta \), topics \( Z \), tags \( T \), and songs \( S \) is computed as follows:

\[
p(S, T, Z, \theta | \mu, \phi) = \prod_{p=1}^{N} p(\theta_p | \alpha) \prod_{i=1}^{M} \left( \sum_{\mu} p(z_i | \theta_p) p(s_i | \mu) \prod_{j=1}^{W} p(t_j | z_i, \phi) \right)
\]

Intuitively, for each user profile, \( \theta_p \) represents the the user’s taste in music, reflected in previous songs that the user has liked and their associated tags describing various characteristics of those songs.

As we will show, by including the tags in the model and sharing common topics between tags and songs, we can personalize the retrieval method. If the tags were not included in the model, although it is possible to recommend songs for a playlist, it is not possible to retrieve songs for a given query. Similarly modeling songs individually, as in the original LDA model, is useful for song retrieval, but without a model that includes users, song retrieval cannot be personalized for users.

**Inference** As the exact inference is not possible for learning the parameters of our model, we use variational message passing for approximate inference. In our implementation we used Infer.Net (Minka et al. 2012) which provides a framework for running Bayesian inference in graphical models and contains various inference algorithms.

### 3.1 Topics

The dataset used for our experiments contains 28,963 user-contributed playlists from “Art of the Mix” website\(^1\) 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 73,045 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. Table 3.1 presents a random sample of topics generated by our model when it is trained for \( k = 50 \) topics. For each of these topics, top 10 tags are shown.

![Graphical representation of the personalized retrieval model](image)

**Figure 2:** Graphical representation of the personalized retrieval model

### 3.2 The Retrieval Method

This section describes application of our model for personalized text-based music retrieval. Consider a user’s profile \( p \) with \( n \) songs. Let \( q = \{t_1, t_2, ..., t_w\} \) represent the user’s query consisting of \( w \) terms \( t_i \). The goal of the retrieval algorithm is to rank songs based on their relevance to the query while considering the user’s general taste in music reflected in their preference history.

We used a language modeling approach for retrieving songs. For a song \( s \), query \( q \) and user \( p \), \( p(s | q, p) \) is computed and is used as a ranking score. According to Bayes rule:

\[
p(s | q, p) \propto p(q | s, p)p(s | p)
\]

After we get the posterior estimates of \( \theta, \mu, \) and \( \phi \), shown as \( \hat{\theta}, \hat{\mu}, \) and \( \hat{\phi} \), we have:

\[
p(\hat{s} | q, p, \hat{\theta}, \hat{\mu}, \hat{\phi}) \propto p(q | \hat{s}, p, \hat{\theta}, \hat{\mu}, \hat{\phi}) p(\hat{s} | p, \hat{\theta}, \hat{\mu})
\]

Where, \( p(s | p, \hat{\theta}, \hat{\mu}) \) is computed as follows:

\[
p(\hat{s} | p, \hat{\theta}, \hat{\mu}) = \sum_{j=1}^{k} p(s | z_j) p(z_j | p) = \sum_{j=1}^{k} \hat{\mu}_{s,j} \hat{\theta}_{j,p}
\]

In equation 5, \( p(q | s, p, \hat{\theta}, \hat{\mu}, \hat{\phi}) \) can be computed as:

\[
p(q | s, p, \hat{\theta}, \hat{\mu}, \hat{\phi}) = \prod_{t_i \in q} p(t_i | s, p, \hat{\theta}, \hat{\mu}, \hat{\phi})
\]

\(^1\)http://www.artofthemix.org/
According to the graphical representation of our model shown in Figure 2:

\[
p(t_i|s, p, \hat{\theta}, \hat{\mu}, \phi) = \sum_{j=1}^{k} p(t_i|z_j, \phi) \cdot p(z_j|s, p, \hat{\theta}, \hat{\mu})
\]

\[
= \sum_{j=1}^{k} \phi_{i,j} \cdot p(z_j|s, p, \hat{\theta}, \hat{\mu})\tag{8}
\]

\[
p(z_j|s, p, \hat{\theta}, \hat{\mu}) = \frac{p(z_j, s, p)}{p(s, p)} = \frac{p(s|z_j) \cdot p(z_j|p)}{\sum_{j=1}^{k} p(s|z_j) \cdot p(z_j|p)}
\]

\[
= \frac{\mu_{s,j} \cdot \theta_{j,p}}{\sum_{j=1}^{k} \mu_{s,j} \cdot \theta_{j,p}}\tag{9}
\]

Using equations 6 and 7, equation 5 can be simplified as follows:

\[
p(s|q, p, \hat{\theta}, \hat{\mu}, \phi) \propto \left(\sum_{j=1}^{k} (\mu_{s,j}\theta_{j,p})\right) \cdot \prod_{t_i \in q} \sum_{j=1}^{k} \phi_{i,j} \cdot \mu_{s,j} \cdot \theta_{j,p}\tag{10}
\]

\textbf{Smoothing} As suggested in (Wei and Croft 2006), and also confirmed in our evaluations, the retrieval performances can be improved if term probabilities computed in equation 8 are further smoothed by linearly combining them with term probabilities computed by the Bayesian smoothing using Dirichlet priors (Zhai and Lafferty 2001): This model computes the probability of \(t_i\) given song \(s\), shown as \(p_{bd}(t_i|s)\), as follows:

\[
p_{bd}(t_i|s) = c(t_i; s) + \delta p(t_i|c)\sum_{t \in s} c(t; s) + \delta \tag{11}
\]

Where \(c(t_i; s)\) is the frequency of tag \(t_i\) in song \(s\) and \(p(t_i|c)\) indicates the probability of \(t_i\) in the whole collection of songs.

For playlist \(p\) and song \(s\), the probability of \(t_i\) computed by our model, shown as \(p_{pm}\), is then linearly combined with \(p_{bd}\):

\[
p(t_i|s, p, \hat{\theta}, \hat{\mu}, \phi) = (1 - \lambda) \frac{c(t_i; s) + \delta p(t_i|c)}{\sum_{t \in s} c(t; s) + \delta} + \lambda p_{pm}(t_i|s, p)\tag{12}
\]

Note that while \(p_{bd}(t_i|s)\) only depends on the song, \(p_{pm}(t_i|s, p)\) depends on both the song and the current playlist.

The smoothing step can improve the results as topic models are too coarse to be used directly for information retrieval and better performance can be achieved by combining them with methods that directly model documents without having latent factors.

### 4 Evaluations

In order to evaluate the performance of our personalized retrieval method, we need to have the songs meta-data, user profiles (or playlists), and user queries. Datasets commonly used for evaluation of collaborative filtering or information retrieval algorithms could not be used for this evaluation as they do not contain all the required information for testing our method.

For the experiments in this section, we used the Art of the mix dataset introduced in section 3.1. We designed two experiments to simulate queries issued by users, which we will discuss later in this section.

In our evaluations, about 8,769 playlists, each containing more than 10 songs that have tags, were used for evaluation. The selected playlists contain 86,262 unique songs. We used 5-fold cross-validation for evaluation. In other words, in each run of the algorithm, 20% of the songs in each playlist were put aside for testing, while the remaining 80% were used for training the model. Additionally, very infrequent songs, i.e., those that have been appearing in less than 10 playlists, were removed from the test set and were added to the training data. For each playlist \(p\), hold-out song \(s\), and query \(q\), each of the competing algorithms provides a ranking over all of the 86,262 candidate songs.

Our model was trained with 50 topics and for 100 iterations. The parameters \(\alpha, \beta, \gamma\) were set to 5, 0.01 and 0.01 respectively. Also, the smoothing parameters \(\delta\) and \(\lambda\) were set to 1000, and 0.7. These values were chosen by tuning against a held-out portion of test data.

<table>
<thead>
<tr>
<th>Topic#22</th>
<th>Topic#24</th>
<th>Topic#23</th>
<th>Topic#39</th>
<th>Topic#41</th>
<th>Topic#1</th>
</tr>
</thead>
<tbody>
<tr>
<td>electronic</td>
<td>alternative</td>
<td>soul</td>
<td>rock</td>
<td>metal</td>
<td>rock</td>
</tr>
<tr>
<td>electronica</td>
<td>indie</td>
<td>rnb</td>
<td>comedy</td>
<td>rock</td>
<td>britpop</td>
</tr>
<tr>
<td>alternative</td>
<td>rock</td>
<td>pop</td>
<td>alternative</td>
<td>industrial</td>
<td>british</td>
</tr>
<tr>
<td>experimental</td>
<td>shoegaze</td>
<td>dance</td>
<td>funny</td>
<td>hardcore</td>
<td>indie</td>
</tr>
<tr>
<td>ambient</td>
<td>upbeat</td>
<td>electronic</td>
<td>punk</td>
<td>alternative</td>
<td>jazz</td>
</tr>
<tr>
<td>indie</td>
<td>nostalgia</td>
<td>rap</td>
<td>pop</td>
<td>metalcore</td>
<td>swing</td>
</tr>
<tr>
<td>rock</td>
<td>amazing</td>
<td>funk</td>
<td>indie</td>
<td>dance</td>
<td>alternative</td>
</tr>
<tr>
<td>chillout</td>
<td>pop</td>
<td>chillout</td>
<td>fun</td>
<td>german</td>
<td>pop</td>
</tr>
<tr>
<td>psychedelic</td>
<td>punk</td>
<td>jazz</td>
<td>quirky</td>
<td>pop</td>
<td>oldies</td>
</tr>
<tr>
<td>pop</td>
<td>noise</td>
<td>americana</td>
<td>silly</td>
<td>punk</td>
<td>beatles</td>
</tr>
</tbody>
</table>
4.1 Experiment I
In experiment I, we used song genre as the "query" and evaluate the retrieval algorithms in their ability in retrieving songs that match the queried genre as well as the user's preferences in music. For songs in our database, genre information was retrieved from Yahoo! music directory. The genre feature in our database consists of 115 classes. The songs in the test data with no match in Yahoo! music directory were removed from the test data and were added to the training data.

4.2 Experiment II
In this experiment, the "query" for each test song was a randomly picked member of its frequent tags set. For example, suppose in a given playlist p, song s is held-out for evaluation. If the set of popular tags for this song is \( T = \{ "rock", "happy", "80s" \} \), then either "rock," "happy," or "80s" was used as a simulated query issued by the user.

4.3 Evaluation Metrics
A personalized retrieval method should assign higher ranks to items that are relevant to the query and also match the user's taste. We used two metrics for evaluation of the retrieval algorithms. The first metric computes the average precision of retrieved items with respect to the query. In experiment I, a relevant song has the same genre as the queried genre class. In experiment II, a relevant song has the query term in its top tags. The average precision at different levels was used to compare the competing algorithms.

To compare different algorithms in their ability to retrieve personalized songs, we look at how well they recover the held-out songs in the test data. For each removed song, its rank in the overall recommendation list is recorded as top recommendations are more valuable for the user. The results for the 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: \( h(N) = |p \in testset : s_p \in R_N(p)|/|testset| \).

4.4 Baseline Methods
The following are the baseline methods we use in our experiments.

**LDA** For each query term \( t_i \), we use equation 2 to compute the conditional probability of \( t_i \) given each of the candidate songs. The model was trained with 50 topics, with \( \alpha \) and \( \beta \) parameters set to 5 and 0.01 respectively. We used the same approach as in section 3.2 for smoothing the probabilities. The smoothing parameters \( \delta \) and \( \lambda \) were set to the 1000 and 0.7 respectively.

**Bayesian Smoothing with Dirichlet Prior** We used Bayesian Smoothing with Dirichlet prior as another baseline language model. Using this model, for a song \( s \) and a query \( Q \), \( p(Q|s) \) is computed according to equation 11. In our evaluations, the \( \delta \) parameter was set to 1000.

**User-Based kNN** User-based \( k \)NN was used as another baseline in our evaluations. The number of neighbors was set to 10 for this evaluation. This approach ignores the user's query and recommend songs just according to the user's history of preferences.

4.5 Analysis of the Results
Table 2 presents the average precision at different levels for experiment I. The average precisions of our personalized retrieval model and the LDA model are very similar at different levels while Bayesian smoothing with Dirichlet priors has slightly better performance. User-based \( k \)NN is performing much worse than the other methods for all the levels. On the other hand, the hit ratio graph for this experiment, shown in Figure 3, indicates that user-based \( k \)NN has the highest hit ratio at all levels. Personalized retrieval has much better hit-ratio than the other two algorithms. Based on average precision and hit ratio metrics, although user-based \( k \)NN recommends songs that match user's taste (captured by hit ratio metric), but the recommended songs do not necessarily match the issued queries resulting in low average precision. Personalized retrieval model is able to keep the average precision almost as high as LDA and Bayesian smoothing model while providing more personalized results as evident by the higher hit ratio.

Table 3 presents the average precision for experiment II. Again, user-based \( k \)NN has the lowest average precision, while the other three algorithms give very similar results. Figure 4 shows the hit ratio at different levels of recommendation for all four methods. Our method achieves the highest hit ratio (at higher ranks).

5 Related Work
Several mixture models have been studied and applied for collaborative filtering. The Bayesian clustering, which is the
Table 2: Average Precision according to the query for experiment I.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>AP at 5</th>
<th>AP at 10</th>
<th>AP at 15</th>
<th>AP at 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Personalized Retrieval Model</td>
<td>0.0032</td>
<td>0.0063</td>
<td>0.0092</td>
<td>0.0120</td>
</tr>
<tr>
<td>Bayesian Smoothing with Dirichlet priors</td>
<td>0.0033</td>
<td>0.0066</td>
<td>0.0100</td>
<td>0.0133</td>
</tr>
<tr>
<td>LDA</td>
<td>0.0030</td>
<td>0.0060</td>
<td>0.0090</td>
<td>0.0121</td>
</tr>
<tr>
<td>User-based kNN</td>
<td>0.0001</td>
<td>0.0002</td>
<td>0.0003</td>
<td>0.0004</td>
</tr>
</tbody>
</table>

Table 3: Average Precision according to the query for experiment II.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>AP at 5</th>
<th>AP at 10</th>
<th>AP at 15</th>
<th>AP at 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Personalized Retrieval Model</td>
<td>0.0032</td>
<td>0.0060</td>
<td>0.0087</td>
<td>0.0111</td>
</tr>
<tr>
<td>Bayesian Smoothing with Dirichlet priors</td>
<td>0.0033</td>
<td>0.0067</td>
<td>0.0100</td>
<td>0.0133</td>
</tr>
<tr>
<td>LDA</td>
<td>0.0033</td>
<td>0.0067</td>
<td>0.0100</td>
<td>0.0133</td>
</tr>
<tr>
<td>User-based kNN</td>
<td>0.0001</td>
<td>0.0002</td>
<td>0.0003</td>
<td>0.0004</td>
</tr>
</tbody>
</table>

Figure 4: Experiment II - Hit ratio for different number of recommendations.

Adaptations or extensions of the aforementioned mixture models have been applied for different problems. In (Marlin 2004), a generative graphical model was introduced for rating-based collaborative filtering. This model is very similar to LDA but it is specially designed to model rating profiles of users instead of binary interactions. Similarly, a graphical model called session model was introduced in (Zheleva et al. 2010) which captures the listening moods of users. Unlike the LDA model, the session model differentiate different playing sessions in user profiles and assumes that there is a latent mood which guides the choice of songs. Thus, beside the overall tastes of users, another latent variable is included in the model to model the mood of the session.

(Boutemedjet and Ziou 2008) proposes a graphical model for context-aware recommendation of visual documents. Their system captures user needs, context and the visual documents in a unified model.

In (Haider et al. 2012), a generative probabilistic model of sessions was introduced. The model creates clusters of similar sessions and uses contextual session information such as time, referrer domain, and link locations to assign a session probabilistically to multiple clusters.

6 Conclusion

We presented a generative hierarchical probabilistic model for personalized retrieval. Although in this paper, we only explored the application of our model for music retrieval, it could be applied for any retrieval problem where user profiles and items textual meta-data are available. Our approach captures the items textual descriptions and the users in a unified model. For a given user and a query, our personalized retrieval method is able to retrieve items that are both relevant to the query and also match the user’s previous preferences.

References


