Query-Driven Context Aware Recommendation

Negar Hariri
DePaul University
College of Computing and Digital Media
Chicago, IL 60604, USA
nhariri@cs.depaul.edu

Bamshad Mobasher
DePaul University
College of Computing and Digital Media
Chicago, IL 60604, USA
mobasher@cs.depaul.edu

Robin Burke
DePaul University
College of Computing and Digital Media
Chicago, IL 60604, USA
rburke@cs.depaul.edu

ABSTRACT

Context aware recommender systems go beyond the traditional personalized recommendation models by incorporating a form of situational awareness. They provide recommendations that not only correspond to a user's preference profile, but that are also tailored to a given situation or context. We consider the setting in which contextual information is represented as a subset of an item feature space describing short-term interests or needs of a user in a given situation. This contextual information can be provided by the user in the form of an explicit query, or derived implicitly.

We propose a unified probabilistic model that integrates user profiles, item representations, and contextual information. The resulting recommendation framework computes the conditional probability of each item given the user profile and the additional context. These probabilities are used as recommendation scores for ranking items. Our model is an extension of the Latent Dirichlet Allocation (LDA) model that provides the capability for joint modeling of users, items, and the meta-data associated with contexts. Each user profile is modeled as a mixture of the latent topics. The discovered latent topics enable our system to handle missing data in item features. We demonstrate the application of our framework for article and music recommendation. In the latter case, the set of popular tags from social tagging Web sites are used for context descriptions. Our evaluation results show that considering context can help improve the quality of recommendations.

Keywords

Context-aware recommendation, Collaborative filtering, Graphical models, Latent Dirichlet Allocation

Categories and Subject Descriptors

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

RecSys'13, October 12–16, 2013, Hong Kong, China.

ACM 978-1-4503-2409-0/13/10 ...$15.00.
http://dx.doi.org/10.1145/2507157.2507187 .

1. INTRODUCTION

Despite the their wide-spread use in many domains, recommender systems still tend to use models that fail to consider the context in which users are interacting with the system. This omission may result in recommendations that, while consistent with general interests of a user, may be inappropriate for the user's current situation or environment. To address this issue, the notion of context-aware recommender systems (CARS) has been introduced.

Although there is no universal agreement on the precise definition of context, one of the most commonly used definitions, proposed by Abowd et al. [1], defines context as: "any information that can be used to characterize the situation of an entity," and by entity they mean "a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves".

A context-aware recommender system uses the available contextual information in addition to the users' profiles in order to generate recommendations best suited to their current situations. Many variations of context-aware recommender systems have been proposed and studied, including those that rely on an explicit representation of contextual factors [3], as well as those that use behavioral cues from users to derive contextual states [5, 13]. A variety of architectural frameworks and ontological categorizations of context aware recommender systems have been considered [2].

Context aware recommender systems may obtain contextual information relevant to a situation or a task in a variety of ways. In this paper, we consider a typical setting in which contextual information is not pre-specified, but rather is obtained in the course of a user's interaction with the system. This information can be *explicitly* acquired by directly asking users or by eliciting it *implicitly* through other means at particular episodes during the interaction. For example, the user of a music recommender system can specify his/her current interest in a specific genre of music by providing that information in the form of a query. Considering both the queried context and the previously established user profile, the system may then recommend songs that best match the user's short-term situational needs as well as the his or her long-term interests. On the other hand, to recommend restaurants to a user, a context-aware restaurant recommender may implicitly consider variables such as time of the day or the location of the user in addition to the user's history of preferences.

We propose a unified probabilistic model that integrates user profiles, item descriptions, and contextual information.

We assume that contextual information can be represented using the same feature space underlying the representation of items. Our model is an extension of the Latent Dirichlet Allocation (LDA) where each user profile is modeled as a mixture of the latent topics. These topics capture the feature associations as well as the item co-occurrence relations in user profiles. Our contextual modeling recommendation framework enables us to systematically optimize the recommendations for the given context. Given a user's profile p, and context q, our model computes the recommendation score for each item i as $p(i|p,q)$ and ranks items based on these scores. We focus on the special case where item features are available (but may not necessarily be complete) and the given context may contain any arbitrary features from the space of items features. Also, we assume that the only data available for each user is the set of items in which he/she has previously expressed interest.

We investigate the application of this model in both music and document recommendation. For document recommendation, we use words in the documents as features. For music recommendation, popular tags are retrieved from social Web sites for all the songs and are used as songs features. For illustration, let us consider an example from the music recommendation domain. Table 1 shows typical user profile (in this case a playing session) consisting of the set of songs that the user likes. A sample of popular tags for each song is also presented. Although users assign different and sometimes contradictory tags to a given song, popular tags are less prone to noise and are descriptive of different characteristics of songs such as genre, mood, theme, era, etc. Set of features common between these songs includes *rock*, *happy*, *2000s*. If the user specifies *80s* as the context, ideal recommendations should match the user's previous interests (*happy* and *rock music*) as well as the queried context (80s music).

The high dimensionality of this feature space can be a problem for many of the popular context-aware recommendation methods such as item-splitting [7], or user-splitting [21]. For the music recommendation example, if context is expressed in terms of a multitude of factors such as genre category, the era, mood, theme, and others, "splitting" approaches can cause sparsity problems as these features may not be available for all the songs. As a result, songs with incomplete feature representations would have low chance of being recommended. More generally, sparsity can be a problem if context vocabulary is large. This situation is especially common when the contextual variables are represented using subsets of item features. One of the advantages of our approach is that the associations between features are captured by the latent factors, and therefore, are integral part of the trained model. As a result, our recommendation approach has a better chance of overcoming the sparsity problem.

2. CONTEXT-AWARE RECOMMENDATION FRAMEWORK

We propose a latent factor probabilistic model which integrates users, items and the context in a single framework. In this paper, we focus on a special setting where context can be represented by a subset of the features associated with items. However, our model can potentially be used for the cases where contextual variables are not related to the items features. For a given user p and a context q, our model computes the recommendation score for each item i as $p(i|p,q)$ and ranks items based on these scores.

Our model is an extension of the *Latent Dirichlet Allocation* (LDA) model [8], 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. LDA has been used for various applications such as document modeling, document classification, and document retrieval [23]. Also, LDA and similar methods such as PLSA [14] can be applied for the task of item recommendation. Looking at user profiles as documents and items as words, the probability of each item for a given user can be computed and used for ranking the items. However, as items features (or the contextual variables) are not part of the model, it is not possible to compute the conditional item probability for a given user and a specific context. Therefore, our extension to the original LDA model is to include the items features in the model.

We model each user as a mixture of topics where topic probabilities for each user are inferred according to the set of items which she/he has previously liked and their features. The topics in our model capture the feature associations as well as the items co-occurrence relations in the users profiles.

We will first review the LDA model and then introduce our extension. Finally, we will present the context-aware recommendation algorithm that uses this extended model.

2.1 Latent Dirichlet Allocation

The plate diagram of the LDA model for modeling documents is shown in Figure 1. In the standard form of graphical models, nodes indicates 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:

θ_i represents the topic distribution for document i
z_{ij} denotes the topic for the j^{th} word in document i
ϕ_k represents the word distribution for topic k.
w_{ij} denotes the j^{th} word in document i.

In this model, θ is a Dirichlet random variable with parameter α:

$$P(\theta|\alpha) = \frac{\Gamma\left(\sum_{i=1}^{k}\alpha_i\right)}{\prod_{i=1}^{k}\Gamma\left(\alpha_i\right)}\theta^{\alpha_1-1}\ldots\theta_k^{\alpha_k-1}$$
$$\text{and } \theta_i \geq 0, \sum_{i=1}^{k}\theta_i = 1 \quad (1)$$

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

Table 1: An example playlist

Artist	Song	A Sample of Popular Tags
Luce	Good day	rock, 2000s, happy, playful, feel good music, music from the oc
Ben Howard	Keep your head up	acoustic, British, folk, indie, singer-songwriter, 2000s, alternative rock
Nada Surf	Always love	Alternative, indie rock, 2000s, happy
Vega 4	Life is beautiful	Alternative, 2000s, indie rock, Greyś anatomy, happy
Imagine Dragons	On top of the world	Alternative, indie rock, 10s, happy, cheerful chill easy listening
State Radio	Right me up	reggae-rock, chill, indie, happiness, fun, mellow, summer

The variable ϕ 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 β 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 ϕ_z a word is picked and is added to the document.

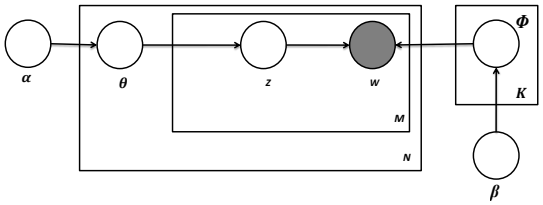

Figure 1: Graphical representation of the LDA model

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 θ given the observed words in the document d. The computed $p(w|d)$ can be used for ranking documents given the query w. Also, if documents corresponds to user profiles and words represent the items, this probability can be to rank items for recommendation.

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

2.2 Context-aware Recommendation Model

The graphical representation of our model, which we call the context-aware recommendation model, is shown in Figure 2. The outermost plate corresponds to users which is replicated N times where N is the number of users. The inner plate, which is replicated M times, correspond to M items that the user likes. The innermost plate, which is replicated W times, indicates the W features of the item.

Each user is modeled as a multinomial distribution over a set of topics where each topic has a distribution over the set of items and features. The random variables in this model are as follows:

θ_i represents the topic distribution for user i
z_{ij} denotes the topic for the j^{th} item in i^{th} user profile.
s_{ij} denotes the j^{th} item in the profile of user i.
t_{ijk} denotes the k^{th} feature for the j^{th} item in the i^{th} user profile.
μ_k represents the item distribution for topic k.
ϕ_k represents the feature distribution for topic k.

Similar to LDA, we assume that θ is a k-dimensional Dirichlet random variable with parameter α, where k is the number of topics.

The variable μ is a $K \cdot U$ random matrix where K is the number of topics and U is the number of unique items. Each row of this matrix is independently drawn from an exchangeable Dirichlet distribution with parameter β and denotes the item distribution for a topic. Similarly, the variable ϕ is a $K \cdot X$ random matrix where X is the size of the feature space. Each row of this matrix is independently drawn from an exchangeable Dirichlet distribution with parameter γ and denotes the feature distribution for a topic.

An item s with W features is assumed to be generated by first choosing a value for z and then sampling the item according to μ_z, and also sampling W features according to ϕ_z. Therefore, it is assumed that each item and its features are generated from the same topic.

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

1. Choose $\theta_p \sim Dir(\alpha)$.
2. Choose $\phi_k \sim Dir(\gamma)$, for each topic k
3. Choose $\mu_k \sim Dir(\beta)$, for each topic k
4. For each of the M items, s_i, in the user's profile:
   (a) Choose $z_i \sim Multinomial(\theta_p)$
   (b) Choose $s_i \sim Multinomial(\mu_{z_i})$
   (c) For each of the W features, t_j, associated with s_i
      i. Choose $t_j \sim Multinomial(\phi_{z_i})$

Given μ and ϕ, the joint distribution of topic mixtures θ, topics Z, features T, and items S is computed as follows:

$$p\left(S, T, Z, \theta | \mu, \phi\right) = \prod_{p=1}^{N} p\left(\theta_p | \alpha\right) \prod_{i=1}^{M} \left(p(z_i|\theta_p) p\left(s_i | z_i, \mu\right) \cdot \prod_{j=1}^{W} p\left(t_j | z_i, \phi\right) \right) \tag{3}$$

Intuitively, for each user profile, θ_p represents the user's taste, reflected in previous items that the user has liked and their associated features.

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

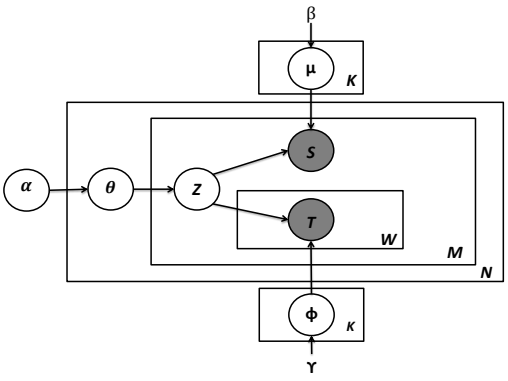

Figure 2: Graphical representation of the context-aware recommendation model

used Infer.Net [20] which provides a framework for running Bayesian inference in graphical models and contains various inference algorithms.

2.3 The Contextual Recommendation Algorithm

This section describes application of our model for context-aware recommendation. Consider a user's profile p with n items. Let $q = \{t_1, t_2, ..., t_w\}$ represent the given context consisting of w features t_i. The goal of the context-aware recommendation algorithm is to provide personalized recommendations for the given context.

Given our model, for an item s, context q and user p, $p(s|q,p)$ is computed and is used as a ranking score. According to Bayes rule:

$$p(s|q,p) \propto p(q|s,p)p(s|p) \tag{4}$$

After we get the posterior estimates of θ, μ, and ϕ, shown as $\hat{\theta}$, $\hat{\mu}$, and $\hat{\phi}$, we have:

$$p\left(s|q,p,\hat{\theta},\hat{\mu},\hat{\phi}\right) \propto p\left(q|s,p,\hat{\theta},\hat{\mu},\hat{\phi}\right) p\left(s|p,\hat{\theta},\hat{\mu}\right) \tag{5}$$

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

$$p\left(s|p,\hat{\theta},\hat{\mu}\right) = \sum_{j=1}^{k} p\left(s|z_j\right) p\left(z_j|p\right) = \sum_{j=1}^{k} \hat{\mu}_{s,j}\hat{\theta}_{j,p} \tag{6}$$

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

$$p\left(q|s,p,\hat{\theta},\hat{\mu},\hat{\phi}\right) = \prod_{t_i \in q} p\left(t_i|s,p,\hat{\theta},\hat{\mu},\hat{\phi}\right) \tag{7}$$

According to the graphical representation of our model shown in Figure 2:

$$\begin{aligned} p\left(t_i|s,p,\hat{\theta},\hat{\mu},\hat{\phi}\right) &= \sum_{j=1}^{k} p\left(t_i|z_j,\hat{\phi}\right) \cdot p\left(z_j|s,p,\hat{\theta},\hat{\mu}\right) \\ &= \sum_{j=1}^{k} \hat{\phi_{i,j}} \cdot p\left(z_j|s,p,\hat{\theta},\hat{\mu}\right) \end{aligned} \tag{8}$$

$$\begin{aligned} p\left(z_j|s,p,\hat{\theta},\hat{\mu}\right) &= \frac{p\left(z_j,s,p\right)}{p\left(s,p\right)} = \frac{p(s|z_j)\cdot p(z_j|p)}{\sum_{y=1}^{k} p(s|z_y)\cdot p(z_y|p)} \\ &= \frac{\hat{\mu_{s,j}} \cdot \hat{\theta_{j,p}}}{\sum_{y=1}^{k} \hat{\mu_{s,y}} \cdot \hat{\theta_{y,p}}} \end{aligned} \tag{9}$$

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

$$\begin{aligned} & p\left(s|q,p,\hat{\theta},\hat{\mu},\hat{\phi}\right) \\ & \propto \left(\sum_{j=1}^{k}\left(\hat{\mu_{s,j}}\hat{\theta_{j,p}}\right)\right) \cdot \prod_{t_i \in q} \sum_{j=1}^{k} \frac{\hat{\phi_{i,j}} \cdot \hat{\mu_{s,j}} \cdot \hat{\theta_{j,p}}}{\sum_{y=1}^{k} \hat{\mu_{s,y}} \cdot \hat{\theta_{y,p}}} \end{aligned} \tag{10}$$

As suggested in [24], and also confirmed in our evaluations, performance can be improved if the probabilities computed in equation 8 are further smoothed by linearly combining them with probabilities computed by the *Bayesian smoothing using Dirichlet priors* [25]; This model computes the probability of t_i given item s, shown as $p_{bd}(t_i|s)$, as follows:

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

Where δ is the smoothing parameter and $c(t_i;s)$ is the frequency of feature t_i in the description for item s and $p(t_i|c)$ indicates the probability of t_i in the whole collection of items descriptions.

For user p and item s, the probability of t_i computed by our model, shown as p_{cr}, is then linearly combined with p_{bd}:

$$p(t_i|s,p) = (1-\lambda)\frac{c(t_i;s) + \delta p(t_i|c)}{\sum_{t \in s} c(t;s) + \delta} + \lambda p_{cr}(t_i|s,p) \tag{12}$$

Note that while $p_{bd}(t_i|s)$ is conditioned only on the item, $p_{cr}(t_i|s,p)$ is conditioned on both the item and the current user.

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

3. EVALUATION OF THE RECOMMENDATIONS

3.1 Datasets

For our evaluations, we used two different datasets that we describe in this section.

Playlists.

This dataset has 28,963 user-contributed playlists from *Art of the Mix* website[1] in January 2003. It consists of 218,261 distinct songs for 48,169 unique artists. The average number of songs per playlist is 19.8 and the average number of artists in playlists is 17.1. Top tags (with frequency of at least 4) 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 or there were no popular tags available for that song. For our evaluations,

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

about 8,769 playlists, each containing more than 10 songs that have tags, were selected. The selected playlists contain 86,262 unique songs.

CiteULike.

Another dataset which we used in our evaluations is the CiteULike *Who-posted-what* dataset[2]. This dataset consists of the set of articles posted by each user, the time and date of posting, and the tags that the user used to post it. The data is available from 2007-05-30 onwards. We pruned the data to contain only users with more than 4 postings and articles with frequency of at least 4. The number of users and articles after pruning were 11,533 and 67,335 respectively. Among the set of articles, we were able to retrieve abstracts for 8,494 articles.

3.2 Experiments

We used 5-fold cross-validation for evaluation. In each run of the algorithm, 20% of the items in each user's profile were put aside for testing, while the remaining 80% items (with their features) were used for training the model. In each run, those items in the test data that have no features were removed from the test set and were added to the train set. For each user's profile p, hold-out item s, and context $q = \{t_1, t_2, ..., t_w\}$ (consisting of w features t_j), each of the competing algorithms provides a ranking over all of the candidate items. As discussed in section 2.3, our model computes the recommendation score for each item i as $p(i|p, q)$ and ranks items based on these scores. The following baseline methods were used in our experiments:

Simple Filtering: For a given context q, this model filters out the items that does not contain q in their list of features. The remaining items are then ranked randomly.

User-Based kNN: This approach ignores the given context and recommend items only based on the users' profiles using the k-nearest-neighbor approach.

LDA: This is a non-personalized, but context-sensitive method, which ignores the users' preference histories and recommends items that best match the given context. We trained the LDA model for the items; assuming that each item is a document and the item features are the words in the document. We used the variational message passing approach, explained in section 2, to learn the parameters of the model. For context $q = \{t_1, t_2, ..., t_w\}$, and each candidate item i, we used equation 2 to compute the conditional probability of each contextual feature t_j given i, shown as $p(t_j|i)$. The same approach as in section 2.3 was applied for smoothing these probabilities. For item i, $p(q|i)$ is then computed as the multiplication of smoothed probabilities for all the contextual features. The computed $p(q|i)$ is used as the recommendation score for item i. The items are then ranked based on these scores.

Bayesian Smoothing: This baseline algorithm is also non-personalized but context-sensitive. It uses Bayesian Smoothing with Dirichlet prior as a language model. Using this model, for an item i and a context $q = \{t_1, t_2, ..., t_w\}$, $p(t_j|i)$ is computed according to equation 11 for each contextual feature t_j. For item i, $p(q|i)$ is then computed as the multiplication of the probabilities for all the contextual features. The computed $p(q|i)$ is used as the recommendation score for item i. The items are then ranked based on these scores.

[2]http://www.citeulike.org/faq/data.adp

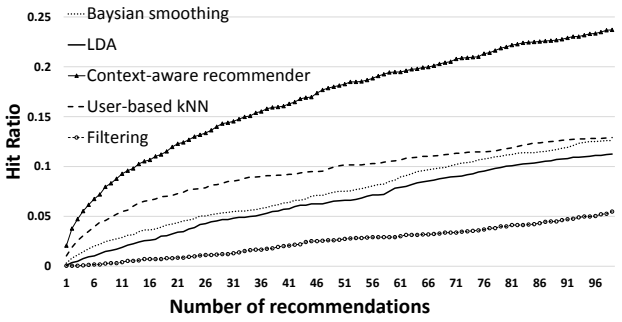

Figure 3: Hit ratio for different number of recommendations for the playlists dataset

The parameters used for each of the algorithms are included in Table 2. We tuned the parameters for each method to achieve the best results.

In our evaluations for the playlists dataset, we assume that each playlist correspond to a user's profile. Also, we used frequent tags for each song as its features. The context for each test song was selected as a randomly-chosen member of its frequent tags set. For example, let song s be a held-out song from playlist p with popular tags T = {"rock", "happy", "80s"}. Any of "rock", "happy", or "80s" could be used as a simulated context when recommending items to include in playlist p. Note that this simulation of context is consistent with our previous assumption that context given to the system is a subset of the features for the desired item.

To compare different algorithms in their ability to provide personalized recommendations for the given context, 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 were evaluated by computing the *Hit Ratio*, which computes the probability that the removed song is recommended as part of the top N recommendations. Formally, we will denote the top N recommendations for a given playlist, p, as $R_N(p)$. If 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 3 compares the hit ratio results of our approach with the other baselines for the first 100 recommendations. As can be seen, our approach achieves significantly better hit ratio at all ranks. At recommendation rank 10, the hit ratio achieved by our method is 3.5% higher than user-based kNN, 5.7% higher than the Bayesian smoothing approach, 7% higher than LDA, and finally 8.4% higher than simple filtering method.

We used the same five-fold cross-validation approach to evaluate our method based on the CiteULike dataset that was earlier introduced. Each user profile contains the postings by that user. The words in each article abstract were

Table 2: Parameters used for each method

Method	Playlists dataset	CiteULike dataset
Context-aware recommender	topics = 50, iterations = 100, α = 5, β = 0.01, γ = 0.01, δ= 1000, λ= 0.7	topics = 70, iterations = 150, α = 3.57, β = 0.01, γ = 0.01, δ= 1000, λ= 0.7
User-based kNN	Number of neighbors=10	Number of neighbors=30
Bayesian smoothing	δ= 1000	δ= 1000
LDA	topics = 50, iterations = 100, α = 5, β = 0.01, δ= 1000, λ= 0.7	topics = 70, iterations = 150, α = 3.57, β = 0.01, δ= 1000, λ= 0.7
Simple filtering	-	-

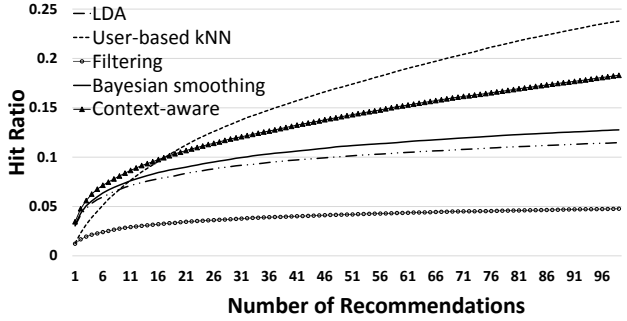

Figure 4: Hit ratio for different number of recommendations for the CiteULike dataset

used as the article features. The context for each test article was selected as the set of tags that the user used to post that article. Note that these tags were not used for training the model as our model is trained based on user profiles and article abstracts.

Figure 4 shows the hit ratio results for this experiment. Our approach achieves the best hit ratio in higher ranks (less than 17) in comparison to the other methods.

The hit ratio of our approach at rank 5 is 2.2% higher in comparison to user-based kNN (that is an improvement of 50%). Also at this rank, the hit ratio of the context-aware recommender is 0.8% (13% improvement), 1% (17% improvement), and 4.5% (197% improvement) higher in comparison to Bayesian smoothing, LDA, and simple filtering method respectively.

4. ANALYSIS OF THE TOPICS

The latent topics in our model can capture the co-occurrence relations among the features of items as well as the associations among items. This section describes a preliminary analysis of some of these associations.

Table 3 presents a few examples of the generated topics for the playlist dataset, showing the top 10 tags, those with the highest probability given the topic. These topics have a clear connection to the genre and, in some cases, the mood of a song. For example, frequent terms in topic# 3 are related to R&B/soul music while topic#4 are mostly describing quirky / humorous songs.

The topics learned from our model can also capture the associations between songs, or similarly, the artists. Using last.fm API, the similar artist for any given artist can be retrieved. We selected the set of 244 artists with at least 50 songs in our dataset of playlists. For each of these artists, the set of similar artists were retrieved from last.fm. We used the similarity data from last.fm as the ground truth and evaluated our model as a representation of artist similarity.

For a model with k topics, each song can be represented as a k-dimensional vector where the j^{th} dimension of the vector represents the probability of the song given the j^{th} topic. After normalizing the song vectors to unit length, they were transformed into binary representation such that each dimension is set to one if its value in the normalized song vector is greater than a pre-specified threshold (that was 0.05 for 50 topics). The elements with the value of one are thereby the *dominant* topics for each song.

We represent each artist as an aggregate of his/her songs vectors. An artist vector representation is generated as the sum of all the binary vectors of the songs for that artist. We compute the similarity of any two artists as the cosine similarity of their corresponding vectors. The artist similarity graph can then be generated. In this graph nodes correspond to the artists and the weight of each edge connecting two artists indicates their cosine similarity.

Given the similarity data from last.fm as the ground truth, the precision of artists' neighbors (in the artist graph) at level X can be computed as the ratio of the number of edges having weight of at least X that are correct to the total number of edges having weight greater than X. An edge connecting artists a and b is counted as correct if b is listed as a similar artist for a (or the reverse) in the last.fm data. Figure 5 shows the precision of artists' neighbors at different thresholds. Figure 6 shows the number of links at each threshold level (> 0.5). As can be seen in Figure 5, the precision of correct neighbors increases by increasing the similarity threshold. When the similarity threshold is equal to zero (that is when the edges are selected randomly), the precision is as low as 5.8%. Precision reaches to 60% when similarity of the artists are greater than 0.95 (which includes 138 links) showing the correlation between topic-based similarities and the true artist neighborhood. Note that for larger threshold values the number of edges become very small (at threshold level of 0.99, only 2 links remains) and the computed precision is not representative of true precision in a larger dataset. so, we did not include them in the precision graph.

5. RELATED WORK

Several researchers have previously investigated the use of contextual information in various recommender systems applications. In [19] a context-aware playlist recommendation system is proposed which provides users opportunity to browse their music by mood. This system uses a combina-

Table 3: A sample of topics generated by the context-aware recommendation model for the playlists dataset

Topic#1	Topic#2	Topic#3	Topic#4	Topic#5	Topic#6
electronic	alternative	soul	rock	metal	rock
electronica	indie	rnb	comedy	rock	britpop
alternative	rock	pop	alternative	industrial	british
experimental	shoegaze	dance	funny	hardcore	indie
ambient	upbeat	electronic	punk	alternative	jazz
indie	nostalgia	rap	pop	metalcore	swing
rock	amazing	funk	indie	dance	alternative
chillout	pop	chillout	fun	german	pop
psychedelic	punk	jazz	quirky	pop	oldies
pop	noise	americana	silly	punk	beatles

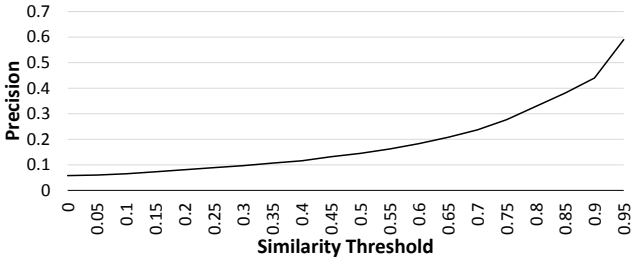

Figure 5: Precision at different similarity levels

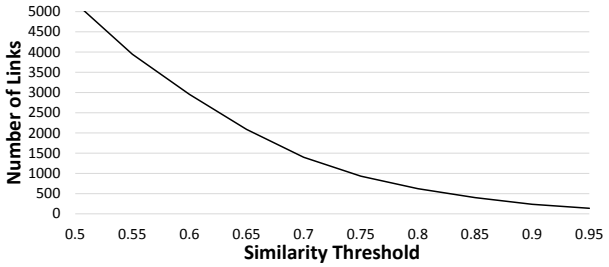

Figure 6: Number of links at different similarity levels

tion of audio and lyric content to classify music by mood. Our approach is different from [19] as the contextual information is not pre-specified in our system.

The system described in [6] focuses on social context and proposes Poolcasting as a method to customize musical sequences for groups of listeners. [11] 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. Unlike [6], the focus of this paper is not on capturing and incorporating the social context. The approach presented in this paper is different from [11] as we are assuming that contextual information is obtained as part of a user's interaction with the system.

In [17] a method is introduced for learning song transition probabilities from audio features extracted from songs played in professional radio station playlists. These transition probabilities are used to compute the similarity of different songs. Given a seed song, the playlist is generated based on songs similarities and also a tag cloud (with 360 tag) that the user is able to manipulate to guide the playlist generation process. A content-based *autotagger* is used to tag all the tracks in the database and to provide the similarity of the tag cloud for each track and the given tag cloud by the user. Unlike [17], the queried features in our system are not limited to a fixed set of tags, as a result the user are not forced to explore the tag cloud to express their interests. Another major difference is that, the recommendations for the next song are generated based on *all* the songs in the playing session or the user's profile as opposed to a single seed song.

There exist fair amount of research work on modeling music playlists. A latent factor model of playlists is proposed in [4]. The model captures the temporal dynamics in the system, such as changes of music type by the time of the day, or similarity of adjacently played tracks. For a given playlist, the model can predict the probability distribution of the next song. In [9] playlists are modeled as Markov chains in some latent space and each song is represented as one or multiple points in this space. The proposed method called Logistic Markov Embedding (LME) can capture the sequential and directed nature of playlists and has been evaluated for the task of playlist prediction. While context is not part of the model in [4] and [9], we are using the given context to adapt the recommendations for the most recent preferences of the users.

Several mixture models have been studied and applied for collaborative filtering. The Bayesian clustering, which is the simplest form of mixture models, discovers latent clusters of users with similar preferences and assumes that each user belongs to a single cluster. The aspect model also finds latent factors representing user clusters but unlike Bayesian cluster models, it assumes that different ratings (or preference feedbacks) of the same user can be explained by different latent causes. The LDA model [8] introduced in Section 2.1 is a generative model in which the proportion of latent factors (also called topics) for each user profile is a random random variable with Dirichlet prior. [16] provides a survey and comparison of various mixture models including Bayesian Clustering [10], the Aspect Model [15], the Flexible Mixture Model [22], and the Joint Mixture Model [22] for collaborative filtering. Adaptations or extensions of the aforementioned mixture models have been applied for a variety of problems. In [18], 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 [26] which captures the listening moods of users. Unlike the LDA model, the session model differentiate different playing sessions in users 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. In [12], a generative probabilistic model of web browsing 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

This paper presented a novel approach for context-aware recommendation. We introduced a unified probabilistic latent factor model that integrates user profiles, item representations, and contextual information. We discussed how our model can be used to compute the conditional probability of each item given the user profile and the additional context.

The findings reported in this paper introduce numerous additional questions and areas of future work. In this paper, we evaluated our model for a special setting in which contextual information is represented as a subset of feature space for the items. However, our proposed model can be extended to handle other types of observable contextual information. In our future work, we plan to generalize this framework and evaluate it in application domains where the representation of context is unrelated to the item features.

7. REFERENCES

[1] G. Abowd, A. Dey, P. Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and context-awareness. *Handheld and Ubiquitous Computing*, 1707(2):304–307, 1999.

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

[3] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Proceedings of the ACM conference on Recommender Systems*. ACM, 2008.

[4] N. Aizenberg, Y. Koren, and O. Somekh. Build your own music recommender by modeling internet radio streams. In *Proceedings of the 21st international conference on World Wide Web*, pages 1–10. ACM.

[5] S. S. Anand and B. Mobasher. Contextual recommendation. In *Lecture Notes In Artificial Intelligence*, volume 4737, pages 142–160. Springer-Verlag, Berlin, Heidelberg, 2007.

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

[7] L. Baltrunas and F. Ricci. Context-based splitting of item ratings in collaborative filtering. In *Proceedings of the third ACM conference on Recommender systems*. ACM, 2009.

[8] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *Journal of machine Learning Research*, 3:993–1022, 2003.

[9] S. Chen, J. L. Moore, D. Turnbull, and T. Joachims. Playlist prediction via metric embedding. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 714–722. ACM, 2012.

[10] Y.-H. Chien and E. George. A bayesian model for collaborative filtering. In *Proceedings of the 7th International Workshop on Artificial Intelligence and Statistics*, 1999.

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

[12] P. Haider, L. Chiarandini, U. Brefeld, and A. Jaimes. Dynamic contextual models for user interaction on the web. In *ECML/PKDD Workshop on Mining and Exploiting Interpretable Local Patterns (I-PAT)*, 2012.

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

[14] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of Uncertainty in Artificial Intelligence*, 1999.

[15] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 688–693, 1999.

[16] R. Jin, L. Si, and C. Zhai. A study of mixture models for collaborative filtering. *Information Retrieval*, 9(3):357–382, 2006.

[17] F. Maillet, D. Eck, G. Desjardins, and P. Lamere. Steerable playlist generation by learning song similarity from radio station playlists. In *ISMIR*, pages 345–350. International Society for Music Information Retrieval, 2009.

[18] B. Marlin. Collaborative Filtering: A Machine Learning Perspective. Master's thesis, University of Toronto, 2004.

[19] O. Meyers. A mood-based music classification and exploration system. Master's thesis, Massachusetts Institute of Technology, 2007.

[20] T. Minka, J. Winn, J. Guiver, and D. Knowles. Infer.NET 2.5, 2012. Microsoft Research Cambridge. http://research.microsoft.com/infernet.

[21] A. Said and E. De Luca. Inferring contextual user profiles - improving recommender performance. In *Context-aware Recommender Systems Workshop at Recsys11*. ACM, 2011.

[22] L. Si and R. Jin. Flexible mixture model for collaborative filtering. In *Proceedings of the international conference on machine learning*, pages 704–711. AAAI Press, 2003.

[23] A. L. Wang. An industrial-strength audio search algorithm. In *Proceedings of the 4th Symposium Conference on Music Information Retrieval*, pages 7–13, 2003.

[24] X. Wei and B. W. Croft. Lda-based document models for ad-hoc retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 178–185, New York, NY, USA, 2006. ACM Press.

[25] C. Zhai and J. D. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In W. B. Croft, D. J. Harper, D. H. Kraft, and J. Zobel, editors, *Proceedings of the annual international ACM SIGIR conference on Research and development in information retrieval*, pages 334–342. ACM, 2001.

[26] E. Zheleva, J. Guiver, E. Mendes Rodrigues, and N. Milić-Frayling. Statistical models of music-listening sessions in social media. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 1019–1028. ACM, 2010.