Knowledge-Based Systems 000 (2018) 1–8

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Incorporating temporal dynamics into LDA for one-class collaborative filtering☆

Haijun Zhang[a], Xiaoming Zhang[b,*], Zhiyong Tian[a], Zhenping Li[a], Jianye Yu[a], Feng Li[a]

[a] School of Information, Beijing Wuzi University, Beijing 101149, China
[b] State Key Laboratory of Software Development Environment, Beihang University, Beijing 100191, China

ARTICLE INFO

Article history:
Received 26 June 2017
Revised 20 February 2018
Accepted 24 February 2018
Available online xxx

Keywords:
One-class collaborative filtering
Latent dirichlet allocation
Temporal dynamics

ABSTRACT

In the one-class collaborative filtering (OCCF) scenario, the elements of the user-item rating matrix consist each take one of only two values: either "like" or unknown. Previous methods for solving the OCCF problem can be roughly categorized into content-based methods, pointwise methods, and pairwise methods. A fundamental assumption of these approaches is that all missing values in a rating matrix can be treated as "dislike". However, this assumption may not hold because the missing values are not always negative. Sometimes users do not give positive feedback on an item simply because they are not familiar with it rather than because they dislike it. In addition, content-based methods usually require textual information on the items. In many cases, however, sufficient textual information is not available; therefore, content-based methods are not applicable. Moreover, a user's preference for items usually drifts over time, but the previous methods cannot capture the temporal dynamics of this drift. In this paper, we propose to modify the latent Dirichlet allocation (LDA) model to address the above-mentioned problems. Our method uses only observed rating data to predict users' interests and effectively avoids the issue of data skew. Furthermore, to address the issue that users' preferences for items usually drift over time, we assign a different weight to each rating according to its timestamp when using Gibbs sampling to estimate the parameters of the LDA model. In this way, the temporal dynamics of the user preferences can be captured. We report experiments conducted to evaluate our model. The results show that the proposed model outperforms state-of-the-art approaches for the OCCF problem.

1. Introduction

Recommender systems have arisen from practical reqirements for personalized e-services in many application domains, such as e-government, e-business, e-commerce, e-libraries, e-learning, e-tourism, e-resource services and e-group activities [1]. Collaborative filtering is a typical and popular information filtering technology used in recommender systems [2]. There are two main focuses of research in the context of collaborative filtering recommender systems: one-class collaborative filtering (OCCF) [3–9] and multi-class collaborative filtering (MCCF) [10–21]. In the OCCF scenario, the values of the elements of the user-item rating matrix R can be only either 1 or unknown. A value of 1 associated with an element r_{ui} means that user u provided positive feedback on item i, e.g., "like" on Facebook, "bought" on Amazon, "collect" on Taobao or "follow" on Sina Weibo. In the MCCF scenario, the elements of the user-item rating matrix are multi-valued, and each represents the degree of a user's preference for an item. Collaborative filtering (CF) recommendation algorithms use the known values in the user-item rating matrix to predict unknown values. Based on this, items that users might wish to buy in the future can be recommended to those users. Machine-learning-based methods [10–13], such as matrix factorization [14–20], have achieved great success in solving the MCCF problem. However, such methods often suffer from a severe over-fitting problem when tackling the OCCF problem because the rating data in OCCF are highly imbalanced [5,22]. To alleviate this problem, content-based methods such as CTR [23] and CTR–SMF [24] have been proposed. However, when content information on the items cannot be obtained, these methods are infeasible. Wang et al. [21] proposed a novel diffusion-based recommendation algorithm which calculates the similarity between users and generates a recommendation list for the target user by considering both implicit and explicit feedback data. This algorithm achieves better performance than the baselines, but it

☆ A short version of this paper has appeared as a poster paper in the proceedings of CIKM 2014.

* Corresponding author.
E-mail addresses: zhanghaijun@bwu.edu.cn (H. Zhang), yolixs@buaa.edu.cn (X. Zhang), tianzhiyong@bwu.edu.cn (Z. Tian), lizhenping@bwu.edu.cn (Z. Li), jy.yu@siat.ac.cn (J. Yu), lifeng@bwu.edu.cn (F. Li).

https://doi.org/10.1016/j.knosys.2018.02.036
0950-7051/

requires explicit feedback data. Pointwise methods [4,7] and pairwise methods [3,5,9] use sampling techniques to alleviate the data skew problem in OCCF. Of the available pairwise methods, BPR [5,9] and GBPR [3] have achieved great success. However, in all of these methods, missing values are considered to be negative. Moreover, none of these methods models temporal dynamics to capture the evolution of a user's interests. Some time-dynamic topic models [25,26] have proposed in the text mining field; however, in that scenario, all the words in a document have the same time stamp, whereas in a recommender system, each rating on an item (treated as a word) recorded for a user (treated as a document) has a time stamp. Zhong et al. [27] proposed a novel LDA-based time-aware service recommendation method to recommend web services for users. Their method requires textual descriptions of web services and search keywords entered by users. Zheng et al. [28] split microblog data into N time intervals and applied the LDA model to each of these sub-datasets individually to capture the interest drift of users. However, in the OCCF scenario, the data are very sparse, and splitting the data into N time intervals will exacerbate this sparsity and cause the estimated parameters to be unreliable. Thus, these time-dynamic topic models cannot be applied to solve the OCCF problem. Recently, based on the FISM [29] approach, the TOCCF [30] method was proposed, in which a time-aware weight is introduced into the loss function to capture the interest drift of users. However, this method also requires the sampling of a considerable amount of negative data.

In this paper, we exploit the latent Dirichlet allocation (LDA) model to address the OCCF problem. Compared with our previous work [8], the differences and new contributions of the present paper are as follows. (1) In our previous work [8], an LDA-based collaborative filtering algorithm was proposed, which is presented in Sections 3.1 and 3.2 of this paper. In Section 3.3 of this paper, we extend the previously proposed model [8] to incorporate the temporal dynamics of a user's preferences. Each observed datum is assigned a specific weight according to its rating timestamp for use when estimating the parameters of the LDA model. Because a user's preference for items can drift over time, the modeling of the temporal dynamics of user preferences should be a key concern when designing recommender systems or general customer preference models. (2) In Section 4, four real-world datasets, including MovieLens 100K[1], MovieLens 1M[1], MovieLens 10M[1] and Netflix[2], are used to evaluate the method. Experiments show that our methods, including our previous method [8], generate better recommendations than state-of-the-art methods. (3) In contrast to the seminal work of Blei et al. [31], Our method is specially designed for solving the OCCF problem and uses Gibbs sampling to estimate the parameters of the LDA model. Gibbs sampling provides a simple and extensible means of incorporating more information into the LDA model. In contrast to other conventional LDA-based CF algorithms [23,24,32–34], the proposed method uses only the observed rating scores for items to predict a user's interests. It neither requires content information on the items nor assumes that users prefer items to which they have previously given positive feedback over other items.

2. Related works

In this section, we review the literature on the state-of-the-art approaches proposed to address the OCCF problem. Overall, there are four main types of approaches for the OCCF problem: (1) content-based methods, (2) pointwise methods, (3) pairwise methods, and (4) time-aware methods.

2.1. Content-based methods

When textual content information on items is available, content-based methods can be applied. In such a method, a graph model is used to model the user-item rating scores by treating each item as a document and all items as a corpus. The values of all unknown data are assumed to be negative and are assigned a small value such as 0, whereas positive feedback is assigned a value of 1. Collaborative topic regression (CTR) [23] and collaborative topic regression with social matrix factorization (CTR–SMF) [24] are two representative examples of content-based methods. The CTR–SMF model is an extension of the CTR model and can be used to provide users with better recommendations when the item contents, rating records and social network information are available. However, the necessary textual contents are usually difficult to obtain, and these two models also treat missing data as negative, which is an unreasonable assumption.

2.2. Pointwise methods

In pointwise methods, unknown user-item pairs are considered equivalent to negative feedback and are assigned a rating score of 0. Then, machine learning methods, e.g., weighted low-rank approximations (WLRA) [4], are designed to fit the rating score matrix. Because most of the ratings are negative, two techniques are commonly adopted to alleviate the data skew issue: (1) assigning a low weight to negative data and a higher weight to positive data or (2) sampling the negative data at a low probability.

2.3. Pairwise methods

Pairwise methods can usually achieve better performance on the OCCF problem than pointwise methods. Some representative pairwise methods include the Bayesian personalized ranking-based matrix factorization (BPR) method [5,9] and the group Bayesian personalized ranking (GBPR) method [3]. The BPR method assumes that user u prefers item i to item j if the user-item pair (u, i) is observed and (u, j) is not. Then, it treats pairs of items as the basic units for analysis and maximizes the likelihood of the pairwise preferences over the observed and unobserved items. In comparison with BPR, GBPR uses a stronger constraint, which can be expressed as shown in Eqs. (1) and (2):

$$if \;\; u \in G_i \; and \;\; u \notin G_j \;\; then \;\; r_{G_i,i} > r_{u,j} \tag{1}$$

$$r_{G_i,i} = \frac{\sum_{u \in G_i} r_{u,i}}{|G_i|} \tag{2}$$

where G_i is the group of users who have given positive feedback on item i, $r_{u,j}$ denotes the preference value of user u for item j, and $r_{G_i,i}$ denotes the group preference value of group G_i for item i, which is defined as the average preference value of all users in group G_i for item i, as expressed in Eq. (2).

In GBPR, the sigmoid function $f\left(r_{G_i,i} - r_{u,j}\right) = \frac{1}{1+\exp\left(-r_{G_i,i}+r_{u,j}\right)}$ is used to approximate the probability $\Pr\left(r_{G_i,i} > r_{u,j}\right)$, and $r_{u,j}$ is factorized as a product of two vectors, as expressed in Eq. (3):

$$r_{u,j} = q_u \cdot s_j^T \tag{3}$$

where q_u is the latent vector of user u and s_j is the latent vector of item j. These latent vectors are learned by maximizing the likelihood of the pairwise preferences.

2.4. Time-aware methods

Time-aware One-Class Collaborative Filtering (TOCCF) [30] is a recently proposed time-aware method for solving the OCCF prob-

[1] http://grouplens.org/datasets/movielens/.
[2] http://www.netflixprize.com/.

Please cite this article as: H. Zhang et al., Incorporating temporal dynamics into LDA for one-class collaborative filtering, Knowledge-Based Systems (2018), https://doi.org/10.1016/j.knosys.2018.02.036

Table 1
Requirements of the algorithms.

Model type	Representative algorithm	Requirements	
		Content information	Sampling of negative data
Content-based	CTR	Y	Y
	CTR–SMF	Y	Y
Pointwise	WLRA	N	Y
Pairwise	BPR	N	Y
	GBPR	N	Y
Time-aware	TOCCF	N	Y
	Our method	N	N

lem that is based on the factored item similarity model (FISM) approach [29]. In TOCCF, a weight corresponding to the estimated error is assigned in accordance with each rating's time stamp, and this weighted error is integrated into the loss function. The TOCCF approach is expressed as shown in Eqs. (4)–(7):

$$\hat{r}_{ui} = b_u + b_i + \left(n_u^+ - 1\right)^{-1} \sum_{j \in R_u^+ \setminus \{i\}} p_j q_i^T \tag{4}$$

$$c_{ui} = \frac{1}{t_T - t_{ui} + 1} \tag{5}$$

$$w_{ui} = \begin{cases} c_{ui}, & if \ \ (u, i, t_{ui}) \in T_u \\ 1, & \text{otherwise} \end{cases} \tag{6}$$

$$\begin{aligned} f = & \frac{1}{2} \sum_{u,i \in R} \left(w_{ui} \left\| r_{ui} - \hat{r}_{ui} \right\|^2 \right) \\ & + \frac{\alpha}{2} \left(\|p_i\|^2 + \left\| q_j \right\|^2 + \|b_u\|^2 + \|b_i\|^2 \right) \end{aligned} \tag{7}$$

where $\hat{r}_{ui}$ is the estimated preference value of user u for item i; b_u and b_i are the user and item biases, respectively; n_u^+ is the number of items evaluated by user u with positive feedback; R_u^+ is the set of items evaluated by user u with positive feedback; p_j and q_i are the learned latent factors for items; (u, i, t_{ui}) denotes feedback generated by user u for item i at time t_{ui}; t_T is the largest time stamp (in days) in the training data; and T_u is the set of positive feedback generated by user u. In the loss function, denoted by f, r_{ui} is the true preference value of user u for item i; R denotes the training data, which include the positive and sampled negative data; and α is a regularization parameter. The limitations of TOCCF are that it assumes that unobserved data are negative and it requires a considerable amount of negative data to be sampled to train the model.

Table 1 summarizes the limitations of the algorithms discussed above. Some of them require content information on items, but such textual information typically cannot be obtained. They all treat unobserved data as equivalent to negative feedback, and they require the sampling of a considerable amount of negative data. The number of these sampled negative data is usually a few times larger than the number of positive data. However, most often, users have not provided positive feedback on an item because they are not familiar with it rather than because they actively dislike it. Consequently, these methods will induce deviations.

3. Exploiting LDA to solve the OCCF problem

3.1. Main idea

In most cases, only the user-item rating matrix is available; no item descriptions are available. If we treat a user as a document, an item as a word, and an item's rating score as the occurrence frequency of the corresponding word in the document, then the LDA model can be applied without requiring the items' descriptions. LDA is capable of capturing latent topics from the user-item rating matrix, and the parameters learned via LDA are probability distributions with two inherent characteristics: the parameters are all positive numbers, and their sum is 1. These characteristics make the model more robust to over-fitting even when faced with severe data skew. Based on this idea, we exploit the LDA model to tackle the OCCF problem. Meanwhile, our model also considers temporal dynamics. A user's more recent rating data should be given higher weights when used to predict that user's future interests. In the following, we will first describe how to exploit the LDA model to tackle the OCCF problem and then introduce the integration of temporal information into the LDA model to capture temporal dynamics.

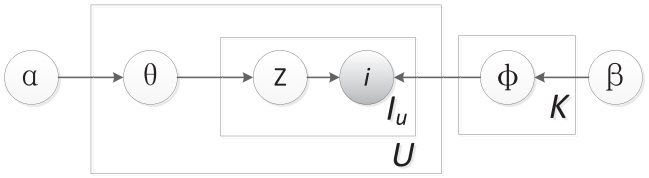

Fig. 1. LDA model for OCCF.

3.2. Basic LDA–OCCF

The latent Dirichlet allocation (LDA) model was proposed by Blei et al. [31] to compensate for the shortcomings of the pLSA model [35]. LDA is an unsupervised probability generative model that can randomly generate the documents that are observed. It overcomes the problem with pLSA by treating the topic mixture weights as a K-parameter hidden random variable rather than a large set of parameters that are directly linked to the training set. It has been shown that LDA is capable of capturing latent topics from a textual corpus [23,24,31–34,36,37]. In the classic LDA scenario, the corpus is a collection of documents, and each document is a collection of words. The corpus can be regarded as a matrix in which each row represents a document, each column represents a word, and each entry in the matrix represents the number of occurrences of the corresponding word in the corresponding document.

In a recommender system, the user-item rating data also form a matrix. Each row of the matrix represents a user, each column of the matrix represents an item, and each rating score represents the degree to which the corresponding user likes the corresponding item. In the traditional OCCF setting, the rating score value is always either 1 or unknown. If we regard a user as a document, an item as a word, and a rating score as the number of occurrences of the corresponding word in the corresponding document, then we can utilize the LDA model to identify a user's latent interests. Here, a user's latent interests are represented by a distribution over topics, and each topic is represented by a distribution over items. We call this proposed model LDA-OCCF. A graphical representation of the proposed model is shown in Fig. 1, where the boxes are "plates" representing replicates. The dimensionality K of the Dirichlet distribution (and thus the number of topics) is assumed to be known and fixed. The document distribution over the topics is represented by a $U \times K$ matrix θ, where U is the number of documents (users). The topic distribution over words is parameterized by a $K \times V$ matrix ϕ, where V is the number of different words (items). The matrices θ and ϕ are generated from two Dirichlet distributions that are defined by the superparameters α and β, respectively. The assignment of topics to items is represented by a $U \times I_u$ matrix Z, where I_u is the number of items evaluated by user u. Our method consists of two steps.

Please cite this article as: H. Zhang et al., Incorporating temporal dynamics into LDA for one-class collaborative filtering, Knowledge-Based Systems (2018), https://doi.org/10.1016/j.knosys.2018.02.036

Step 1. Learn the matrices θ and ϕ, i.e., the user distribution over topics and the topic distribution over items, respectively. The generation process is as follows.

1. For each user u, draw a topic proportion $\theta_u \sim Dirichlet(\alpha)$.
2. For each topic k, draw an item proportion $\phi_k \sim Dirichlet(\beta)$.
3. For each observed user-item pair (u, i),
   a) draw a topic assignment $z_{ui} \sim Mult(\theta_u)$, and
   b) draw an item $i \sim Mult\left(\phi_{z_{ui}}\right)$.

Here, $Dirichlet(\alpha)$ denotes the Dirichlet distribution with parameter α, $Mult(\theta_u)$ denotes the multinomial distribution with parameter θ_u, and z_{ui} denotes the topic assigned to the user-item pair (u, i). We use the Gibbs sampling method to estimate the parameters θ and ϕ. For more information about this method, please refer to [38]. Here, the values of the number of topics K and the superparameters α and β are set in advance.

Step 2. Compute the preference of user u for item i using Eq. (8):

$$r_{ui} = p(i|u) \propto \sum_{k=1}^{K} p(k|u)p(i|k) = \sum_{k=1}^{K} \theta_{uk}\phi_{ki} \tag{8}$$

where p($i|u$) is the probability that item i will occur given user u, θ_{uk} is the value of the user distribution over topics for user u and topic k, and ϕ_{ki} is the value of the topic distribution over items for topic k and item i. From Eq. (8), we can see that r_{ui} has a high value only when θ_{uk} and ϕ_{ki} are both large numbers. This conforms to reality. For example, if user u likes action movies and movie i contains many action scenes, then we can infer that user u has a high probability of liking movie i, or vice versa. Here, the action genre is regarded as a topic. Thus, using Eq. (8) to measure the preference of user u for item i is reasonable. Once each user's preferences for all items have been calculated using Eq. (8), items can be recommended to each user based on the ranking of those items.

The LDA–OCCF model is superior to the traditional models in the following three respects. (1) Unlike in all previous methods, in LDA–OCCF, the values of missing ratings are assumed to be unknown rather than negative, and only observed data are sampled. (2) Unlike the CTR and CTR–SMF models, LDA–OCCF does not require content information on the items. (3) Unlike in pointwise or pairwise methods, the parameters learned in the LDA–OCCF model are probability distributions, and thus, LDA–OCCF is more robust against over-fitting.

3.3. Extension of LDA–OCCF using temporal information

The algorithms discussed above, such as BPR, GBPR, CTR, CTR-SMF and our proposed LDA–OCCF method, are all independent of time, and they do not consider the temporal dynamics of users' interests. In reality, a user's interests often change as time passes. In this section, we extend the LDA–OCCF model to capture and utilize temporal information.

In LDA–OCCF, all observed user-item pairs (u, i) are used to estimate the parameters θ and ϕ, and they are all weighted the same for updating the parameters. However, many datasets, such as MovieLens 100K, MovieLens 1M, MovieLens 10M, and Netflix, all contain rating timestamps. This means that each observed user-item pair (u, i) has time information attached. We can use a triple (u, i, t) to represent that user u gave a positive feedback rating to item i at time t. Given two triples (u, i_1, t_1) and (u, i_2, t_2), if $t_1 < t_2$, it can be concluded with a high level of confidence that item i_2 better reflects the more recent interest of user u than does item i_1. This is because a user's inclinations are constantly evolving, and a user's more recent rating behavior likely better reflects his latest interests. Thus, the triple (u, i_2, t_2) should be assigned a higher weight than the triple (u, i_1, t_1) when both are used to estimate the user's interests. Based on the above analysis, we propose a new sampling method called LDA–OCCF-Temporal for modeling the temporal dynamics of user preferences. The pseudocode flow of the algorithm is shown below.

Algorithm 1 LDA–OCCF-temporal algorithm.

Require: observed user-item rating triples (u, i, t), hyperparameters α and β, number of topics K.
Ensure: multinomial parameters θ and ϕ

```
//initialization
set all count variables n_u^k, n_u, n_k^i, and n_k to zero
for all users u do
    for all triples (u, i, t) do
        sample topic index z_ui = k ~ Mult(1/K)
        increment user-topic count: n_u^k += f(t)
        increment user-topic sum: n_u += f(t)
        increment topic-item count: n_k^i += f(t)
        increment topic-item sum: n_k += f(t)
    end for
end for
//Gibbs sampling over burn-in period and sampling period
while not finished do
    for all users u do
        for all triples (u, i, t) do
            //for the current assignment of k to item i, decrement counts and sums
            n_u^k -= f(t)
            n_u -= f(t)
            n_k^i -= f(t)
            n_k -= f(t)
            sample topic index k~ ~ p(z_ui = k~) acc. to Eq. (9)
            //for the new assignment of z_ui, increment counts and sums:
            n_u^k~ += f(t)
            n_u += f(t)
            n_k~^i += f(t)
            n_k~ += f(t)
        end for
    end for
    //check convergence and read out parameters
    if converged and L sampling iterations since last read out then
        //average the different read-out parameters
        read out parameter set φ according to Eq. (10)
        read out parameter set θ according to Eq. (11)
    end if
end while
```

Eqs. (9)–(11) used in the LDA–OCCF-Temporal algorithm are as follows:

$$p(z_{ui} = k) \propto \frac{n_k^i + \beta_i}{\sum_i n_k^i + \beta_i}\left(n_u^k + \alpha_k\right) \tag{9}$$

$$\phi_{ki} = \frac{n_k^i + \beta_i}{\sum_i n_k^i + \beta_i} = \frac{n_k^i + \beta_i}{n_k + \beta_i} \tag{10}$$

$$\theta_{uk} = \frac{n_u^k + \alpha_k}{\sum_k n_u^k + \alpha_k} = \frac{n_u^k + \alpha_k}{n_u + \alpha_k} \tag{11}$$

Eq. (9) defines how to sample a topic of an item i that has been evaluated by user u. Eqs. (10) and (11) define how to calculate the parameter sets ϕ and θ, respectively. The number of times that item i has been observed with topic k is denoted by n_k^i. The observation count for topic k assigned to items rated by user u is denoted by n_u^k. The number of items evaluated by user u is denoted

Please cite this article as: H. Zhang et al., Incorporating temporal dynamics into LDA for one-class collaborative filtering, Knowledge-Based Systems (2018), https://doi.org/10.1016/j.knosys.2018.02.036

by n_u, and the number of items to which topic k is assigned is denoted by n_k. Detailed derivations of Eqs. (9)–(11) can be found in [38]. The main difference here is that a user is treated as a document and an item is treated as a word. In addition to this difference, the increment applied to the count variables (n_u^k, n_k^i, n_u, and n_k) is represented by the function $f(t)$, which depends on the rating timestamp t. This function denotes the weight with which a rating triple (u, i, t) is used to adjust the user-topic and topic-item distributions, i.e., the parameters θ and ϕ. Previous Gibbs sampling methods [38], including LDA–OCCF, operate independently of time, which means that $f(t) \triangleq 1$.

Because a user's inclinations evolve and, therefore, a user's more recent ratings better reflect his latest interests, we use t_s to denote the time at which the first rating feedback in the dataset was given and t_e to denote the time at which the latest rating feedback was given. Similar to what is done in [20], we split the time span $t_e - t_s$ into m equal intervals, thereby forming m bins. The time interval applied in this paper is chosen to be either 30 days or 90 days, depending on the dataset. The weight of a rating triple (u, i, t) is represented by the function $f(t)$. In this paper, for simplicity, we use a linear function to capture the possible gradual drift of a user's interests, as defined in Eq. (12):

$$f(t) = \frac{b_0 + Bin(t)}{b_0 + m} \tag{12}$$

where b_0 is a positive constant that is used to smooth the slope and $Bin(t)$ denotes the bin into which rating timestamp t is placed and takes a value between 1 and m.

Once the parameters θ and ϕ have been estimated, we can use Eq. (8) to compute each user's preferences for all items. Based on the results, items can be recommended to a user based on the ranking of those items.

4. Experimental results

4.1. Datasets

We used the following real-world datasets in our empirical studies: MovieLens 100K, MovieLens 1M, five subsets of MovieLens 10M and five subsets of Netflix. We will briefly describe these datasets in the following.

The MovieLens 100K dataset contains 100,000 ratings assigned by 943 different users to 1682 movies over seven months, from September 19, 1997, to April 22, 1998. These ratings are on a scale from 1 to 5. We preprocessed the data in four steps. Step 1: Similar to what was done in [3], we sampled only ratings higher than 3 as instances of observed positive feedback and replaced them with values of 1 to simulate one-class feedback. Step 2: We split the sampled data into two parts: one for training and the other for testing. The training set consisted of the ratings from the first six months, and the test set consisted of the ratings from the last month. Step 3: The training data were further split into 6 bins according to their rating timestamps, where each bin had a time span of approximately 30 days. Step 4: For a given triple (u, i, t) in the test data, if user u or item i was not observed in the training data, that triple was deleted from the test data. After these processing steps, we obtained 45,786 training records and 983 test records. These ratings were produced by 777 users for 1387 items with a density of 0.0434, meaning that only 4.34% of the elements in the user-item rating matrix were observed (including both the training data and test data), whereas 95.66% of the elements were missing. The reported evaluation results for the MovieLens 100K dataset are the average values over 5 independent runs.

The MovieLens 1M dataset contains 1,000,209 ratings assigned by 6040 users to 3952 movies over 3 years, from 2000 to 2003. The preprocessing of this dataset was similar to the preprocessing of the MovieLens 100k dataset as described above, except that the training data here were split into 10 bins, each with a time span of approximately 90 days, and the ratings assigned by users during the last 180 days were selected as the test data. After preprocessing, we obtained 570,474 training records and 4625 test records. These ratings were produced by 6036 users for 3531 items with a density of 0.0270. The reported evaluation results for the MovieLens 1M dataset are the average values over 5 independent runs.

Table 2
Datasets used in the experiments.

Dataset	Training data	Test data	Users	Items	Density
ML100K	45,786	983	777	1387	0.0434
ML1M	570,474	4625	6036	3531	0.0270
ML10M-1	364,195	2353	9338	4039	0.0097
ML10M-2	365,824	3879	9364	4185	0.0094
ML10M-3	360,537	3510	9374	4120	0.0094
ML10M-4	374,742	3643	9384	4148	0.0097
ML10M-5	373,840	3345	9361	4043	0.0100
Netflix-1	259,380	51,410	7636	3769	0.0108
Netflix-2	231,486	44,924	7554	3733	0.0098
Netflix-3	254,884	48,586	7638	3766	0.0106
Netflix-4	251,787	48,385	7638	3781	0.0104
Netflix-5	239,204	42,775	7592	3748	0.0099

The MovieLens10M dataset contains 10,000,054 ratings assigned to 10,681 movies by 71,567 users over 13 years, from 1996 to 2009. This dataset was preprocessed in 2 steps. Step 1: We randomly sampled 5 sub-datasets from the MovieLens 10M dataset, labeled ML10M-1, ML10M-2, ML10M-3, ML10M-4, and ML10M-5. Each sub-dataset contained 5000 movies and 10,000 users. Step 2: Preprocessing steps similar to those applied to the MovieLens 100K dataset were applied to each sub-dataset, except that the training data here were split into 50 bins, each with a time span of approximately 90 days, and the test data consisted of the ratings assigned by users during the last 180 days. Detailed statistics of these sub-datasets are shown in Table 2. The reported evaluation results for the MovieLens 10M dataset are the average values over 5 independent runs on each of these 5 sub-datasets individually.

The Netflix dataset contains over 100 million ratings assigned by 480,000 customers to 17,000 movies over approximately 6 years. We also randomly sampled 5 sub-datasets from the Netflix dataset. Each sub-dataset contained 5000 movies and 10,000 users. The preprocessing steps applied to these sub-datasets were similar to those applied to the MovieLens 10M dataset, except that the training data here were split into 20 bins, each with a time span of approximately 90 days, and the test data consisted of the ratings assigned by users during the last 180 days. Detailed statistics of these 5 sub-datasets are also shown in Table 2. The reported evaluation results for the Netflix dataset are the average values over 5 independent runs on each of these 5 sub-datasets individually.

In Table 2, ML100K and ML1M are abbreviations for MovieLens 100K and MovieLens 1M, respectively. Two considerations were taken into account when setting the size of each bin. One is the total time span of the dataset. The other is the supposition that over a span of only 90 days, users' inclinations may not change considerably. Our proposed algorithm uses the training data to learn each user's latest interests and then uses the test data to evaluate the recommendation performance. Time-independent algorithms such as BPR and GBPR cannot utilize the ratings' timestamp information; therefore, each rating in the training data has the same weight when used to estimate the model parameters.

4.2. Evaluation metrics

We assume that users of recommender systems usually check only a few top-ranked items; thus, we use top-k evaluation met-

rics, including the precision, recall, and mean average precision (MAP), to study the OCCF recommendation performance. For each evaluation metric, we first calculate the performance for each user from the test data and then average the results over all test users.

The precision is the proportion of the top recommendation results that are relevant. The recall is the proportion of all relevant results that are included in the top results. These metrics can be computed using Eqs. (13) and (14), respectively.

$$Prec@N = \frac{r}{N} \tag{13}$$

$$Rec@N = \frac{r}{|T|} \tag{14}$$

where r is the number of relevant items in the top-N recommendation list for user u, which means that these r items are, in fact, preferred by user u in the test set. The symbol $|T|$ denotes the number of all preferred items indicated by user u in the test set. Larger precision and recall values indicate that the algorithm achieves a better recommendation performance.

The MAP is widely used in information retrieval to evaluate ranked documents over a set of queries. We use the MAP in this paper to assess the overall performance based on the precision values at different levels of recall on the test set. The MAP is computed as the mean of the average precision (AP) over all users in the test set. The AP for user u is computed by considering each of the preferred items in the ranked list as shown in Eq. (15):

$$AP_u = \frac{\sum_{i=1}^{N} prec(i) \times pref(i)}{\#\ of\ preferred\ items\ of\ user\ u\ in\ test\ set} \tag{15}$$

where i is the position in the ranked list, N is the number of recommended items for which user u did not provide positive feedback in the training set, $prec(i)$ is the precision calculated for a truncated ranked list consisting of the top-ranked items from 1 to i, and $pref(i)$ is a binary indicator that returns 1 when the ith item is preferred by user u or 0 otherwise. The MAP is the mean of AP_u over all test users. The larger the MAP score is, the better the algorithm performs.

4.3. Baselines and parameter settings

We used four classical recommendation algorithms as baselines in our experiments: PopRank, BPR [5], GBPR [3], and TOCCF [30]. PopRank is a basic algorithm for one-class collaborative filtering that ranks items according to their popularity in the training data. As a seminal work on the OCCF problem, BPR [5] has been shown to be far superior to pointwise methods [4,5,7]. Note that we implemented this baseline using the publicly available source code package MyMediaLite [39]. GBPR [3] is a recently proposed method that has outperformed BPR in some experiments [3]. TOCCF [30] is a time-aware method and a strong baseline algorithm.

In the LDA–OCCF-Temporal algorithm, the parameter b_0 in Eq. (12) was tested with values of {0, 5, 10, 20}. The parameters of BPR and GBPR were tuned according to the recommendations in [5] and [3], respectively. To ensure a fair comparison of the algorithms, the number of iterations was set to the same value (10,000) for BPR, GBPR, LDA–OCCF and LDA–OCCF-Temporal. The number of topics in the LDA–OCCF and LDA–OCCF-Temporal algorithms and the dimensionality of the latent vectors (q_u and s_j in Eq. (3)) in the BPR and GBPR algorithms also were all set to the same value of 50. For TOCCF, paper [30] reported the best values of the hyperparameters, and we adopted the corresponding settings as follows. On the MovieLens 100K, MovieLens 1M, and MovieLens 10M datasets, the regularization factor was set to 0.01, and the algorithm was iterated 500 times. On the Netflix dataset, the regularization factor was set to 0.001, and the algorithm was iterated 100 times. On all datasets, the number of factors was set to 20, the learning ratio was set to 0.01, and the number of sampled negative data was 3 times the number of positive data.

We tested the algorithms on a personal computer with a 4-core Intel(R) Core(TM) CPU i5-3470 at 3.20 GHz with 6 GB of main memory, running the Windows 7 operating system.

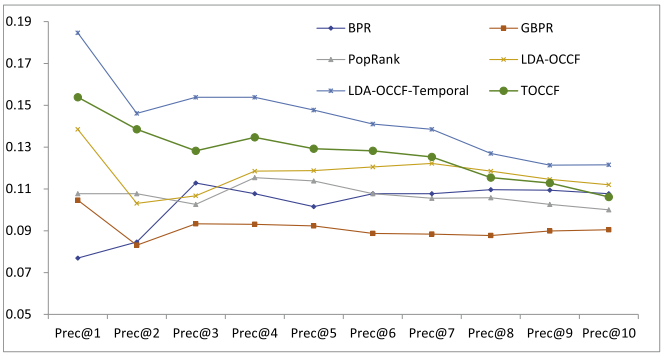

Fig. 2. Precision on MovieLens 100K.

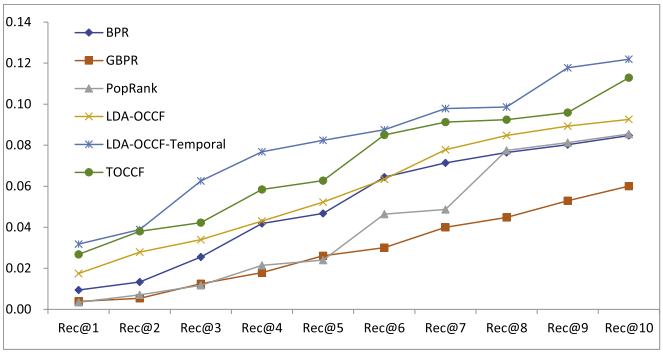

Fig. 3. Recall on MovieLens 100K.

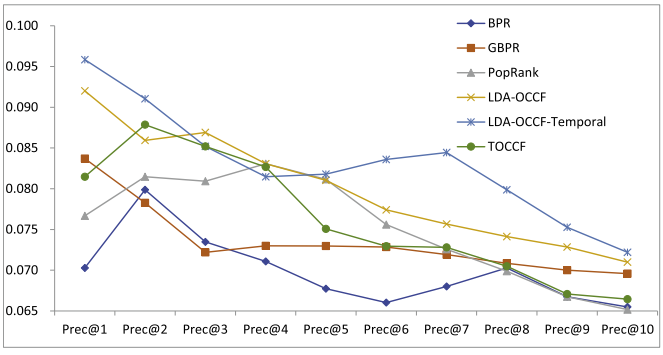

Fig. 4. Precision on MovieLens 1M.

4.4. Experimental results

The evaluation results are shown in Figs. 2–11. We can make the following observations:

1. The BPR, TOCCF, LDA–OCCF and LDA–OCCF-Temporal methods usually perform better than PopRank, which indicates that personalized recommendation is generally superior to popularity-based recommendation.
2. LDA–OCCF usually achieves top-3 performance on all datasets, which indicates that avoiding data bias while modeling only observed data is feasible.

Please cite this article as: H. Zhang et al., Incorporating temporal dynamics into LDA for one-class collaborative filtering, Knowledge-Based Systems (2018), https://doi.org/10.1016/j.knosys.2018.02.036

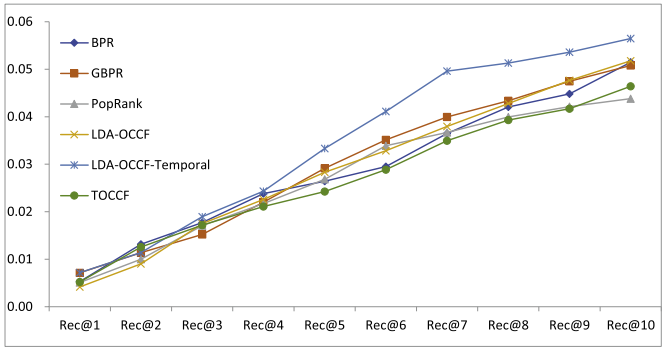

Fig. 5. Recall on MovieLens 1M.

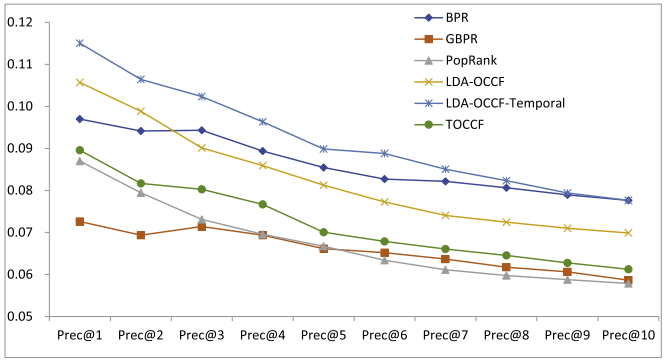

Fig. 6. Precision on MovieLens 10M.

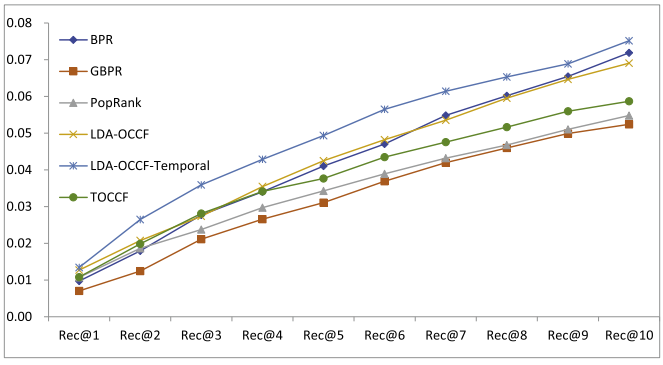

Fig. 7. Recall on MovieLens 10M.

3. LDA–OCCF-Temporal almost always achieves the highest precision, recall and MAP on all datasets. This implies that incorporating temporal information into the model does help to better capture users' latest preferences and thus improves the recommendation performance.
4. On the MovieLens 100K dataset, TOCCF achieves higher precision and recall than all other methods except LDA–OCCF-Temporal, which indicates that incorporating temporal information is useful. However, TOCCF does not achieve better performance on the other datasets, and its time cost is always the largest because it samples too many negative data. Moreover, according to Eq. (4), for each rating r_{ui}, TOCCF must compute the sum of the latent factors of the items rated by user u, which increases its computational burden.
5. Although LDA–OCCF-Temporal and BPR achieve similar precision and recall on the Netflix dataset, the time consumption of LDA–OCCF-Temporal is significantly smaller than that of BPR. This is because LDA–OCCF and LDA–OCCF-Temporal only sample the observed data, whereas BPR samples all pairs of items.

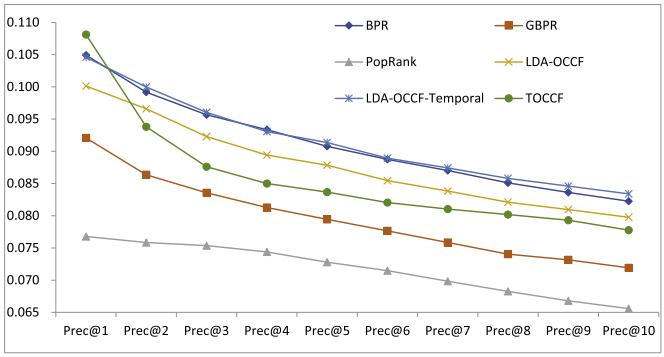

Fig. 8. Precision on Netflix.

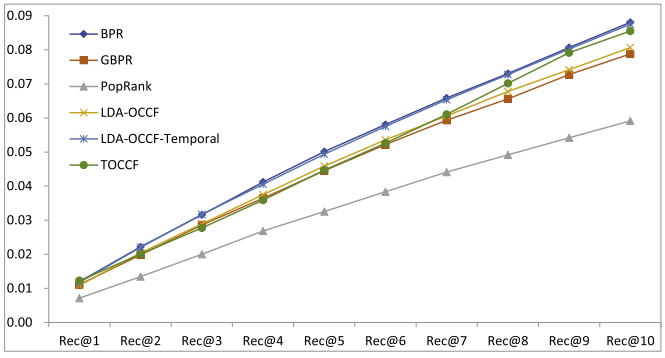

Fig. 9. Recall on Netflix.

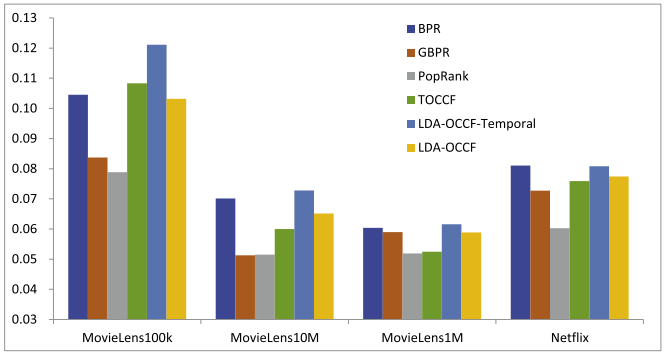

Fig. 10. MAPs of the different methods on all four datasets.

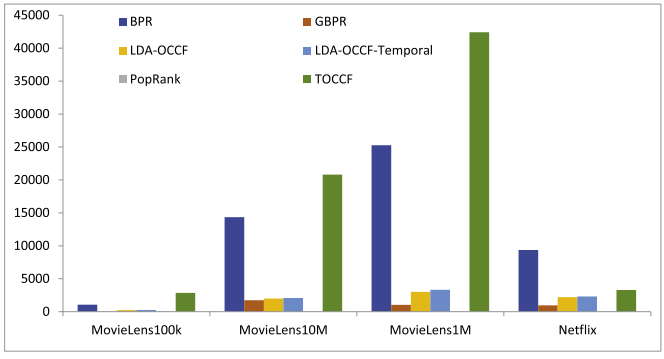

Fig. 11. Run times (in seconds) of the different methods on all four datasets.

From the results, one can also see that GBPR is not significantly superior to PopRank in terms of precision and recall. This is because 10,000 iterations on the training dataset are not suf-

Please cite this article as: H. Zhang et al., Incorporating temporal dynamics into LDA for one-class collaborative filtering, Knowledge-Based Systems (2018), https://doi.org/10.1016/j.knosys.2018.02.036

ficient for GBPR. When the number of iterations is increased to 100,000, GBPR usually achieves higher precision and recall on the datasets than PopRank does. However, it is still inferior to LDA–OCCF and LDA–OCCF-Temporal, and its run time is considerably longer. Thus, it can be concluded that LDA–OCCF and LDA–OCCF-Temporal achieve superior recommendation performance with a lower time cost compared with other current methods.

5. Conclusion

This paper has proposed two algorithms based on the LDA model, named LDA–OCCF and LDA–OCCF-Temporal, to address the OCCF problem. In contrast to previous methods, our algorithms require neither the strong assumption that all missing data are negative nor content information on the items. Furthermore, the proposed LDA–OCCF-Temporal method utilizes rating timestamps to model the drift of users' interests over time. Experiments on several real datasets have shown that our proposed methods achieve better performance than other state-of-the-art algorithms. In the future, we plan to optimize the design of the timestamp-dependent function for tuning the weight of each rating to further improve the LDA–OCCF-Temporal algorithm's performance.

Acknowledgments

We are extremely grateful to Weike Pan, i.e., the author of reference [3], for supplying his code for the GBPR algorithm, which helped us to perform the evaluations more efficiently and more fairly. Dr. Senzhang Wang polished the paper and gave some advice. This work was supported by the National Natural Science Foundation of China (No. 61571052, No. 71771028), the Fund of Beijing Education Commission (No. KM201810037001), a Research Project of the National Bureau of Statistics (No. 2013LY055), and the Fund of Beijing Wuzi University (No. 2017GG03).

References

[1] J. Lu, D. Wu, M. Mao, W. Wang, G. Zhang, Recommender system application developments: a survey, Decis. Supp. Syst. 74 (C) (2015) 12–32.
[2] Z. Xu, F. Zhang, W. Wang, H. Liu, X. Kong, Exploiting trust and usage context for cross-domain recommendation, IEEE Access 4 (2016) 2398–2407.
[3] W. Pan, L. Chen, GBPR: group preference based Bayesian personalized ranking for one-class collaborative filtering[c], in: Proceedings of International Joint Conference on Artificial Intelligence(IJCAI 2013), pp. 2691–2697.
[4] R. Pan, Y. Zhou, B. Cao, N.N. Liu, R. Lukose, M. Scholz, Q. Yang, One-class collaborative filtering[c], in: Proceedings of IEEE International Conference on Data Mining(ICDM 2008), pp. 502–511.
[5] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, BPR: Bayesian personalized ranking from implicit feedback[c], in: Proceedings of Conference on Uncertainty in Artificial Intelligence, AUAI Press, 2012, pp. 452–461.
[6] Y. Li, J. Hu, C.X. Zhai, Y. Chen, Improving one-class collaborative filtering by incorporating rich user information[c], in: Proceedings of ACM Conference on Information and Knowledge Management, CIKM 2010, Toronto, Ontario, Canada, 2010, pp. 959–968.
[7] Y. Hu, Y. Koren, C. Volinsky, Collaborative filtering for implicit feedback datasets[c], in: Proceedings of IEEE International Conference on Data Mining, ICDM, 2008, pp. 263–272.
[8] H. Zhang, Z. Li, Y. Chen, X. Zhang, S. Wang, Exploit latent Dirichlet allocation for one-class collaborative filtering[c], in: Proceedings of ACM International Conference on Information and Knowledge Management(CIKM 2014), pp. 1991–1994.
[9] M. Liu, W. Pan, M. Liu, Y. Chen, X. Peng, Z. Ming, Mixed similarity learning for recommendation with implicit feedback, Knowl. Based Syst. 119 (C) (2017) 178–185.
[10] B. Li, Q. Yang, X. Xue, Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction[c], in: Proceedings of International Joint Conference on Artificial Intelligence(IJCAI 2009), Pasadena, California, Usa, 2009, pp. 2052–2057.
[11] T. Hofmann, Latent semantic models for collaborative filtering[j], ACM Trans. Inf. Syst. 22 (1) (2004) 89–115.
[12] R. Salakhutdinov, A. Mnih, G. Hinton, Restricted Boltzmann machines for collaborative filtering[c], in: Proceedings of 24th Annual International Conferenceon Machine Learning (ICML 2007), pp. 791–798.
[13] H. Zhang, C. Liu, Z. Li, X. Zhang, Collaborative filtering based on rating psychology[c], in: Proceedings of International Conference on Web-Age Information Management(WAIM 2013), Springer-Verlag, 2013, pp. 655–665.
[14] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (8) (2009) 30–37.
[15] H. Ma, H. Yang, M.R. Lyu, I. King, Sorec: social recommendation using probabilistic matrix factorization[c], in: Proceedings of the 17th ACM conference on Information and knowledge management(CIKM), ACM, 2008, pp. 931–940.
[16] S. Funk, Netflix Update: Try This at Home, Tech. Rep., 2006. http://www.sifter.org/~simon/journal/20061211.html.
[17] N. Srebro, T. Jaakkola, Weighted low-rank approximations[c], in: Proceedings of International Conference on Machine Learning(ICML), 2003, pp. 720–727.
[18] P. Cremonesi, Y. Koren, R. Turrin, Performance of recommender algorithms on top-n recommendation tasks[c], in: Proceedings of the Fourth ACM Conference on Recommender Systems(RecSys), ACM, 2010, pp. 39–46.
[19] A. Mnih, R. Salakhutdinov, Probabilistic matrix factorization[c], in: Advances in Neural Information Processing Systems, 2007, pp. 1257–1264.
[20] Y. Koren, Collaborative filtering with temporal dynamics[j], Commun. ACM 53 (4) (2010) 89–97.
[21] X. Wang, Y. Liu, G. Zhang, Y. Zhang, H. Chen, J. Lu, Mixed similarity diffusion for recommendation on bipartite networks[j], IEEE Access 5 (2017) 21029–21038.
[22] H. He, E.A. Garcia, Learning from imbalanced data[j], IEEE Trans. Knowl. Data Eng. 21 (9) (2009) 1263–1284.
[23] C. Wang, D.M. Blei, Collaborative topic modeling for recommending scientific articles[c], in: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD), San Diego, Ca, Usa, 2011, pp. 448–456.
[24] S. Purushotham, Y. Liu, C.C.J. Kuo, Collaborative topic regression with social matrix factorization for recommendation systems[c], in: Proceedings of International Conference on Machine Learning(ICML), 2012, pp. 691–698.
[25] D.M. Blei, J.D. Lafferty, Dynamic topic models[c], in: Proceedings of the 23rd International Conference on Machine Learning(ICML), ACM, 2006, pp. 113–120.
[26] C. Wang, D.M. Blei, D. Heckerman, Continuous time dynamic topic models[j], Uncertain. Artif. Intel. (2008) 579–586.
[27] Y. Zhong, Y. Fan, K. Huang, W. Tan, J. Zhang, Time-aware service recommendation for mashup creation in an evolving service ecosystem[c], in: IEEE International Conference on Web Services, 2014, pp. 25–32.
[28] N. Zheng, S. Song, H. Bao, A temporal-topic model for friend recommendations in chinese microblogging systems[j], IEEE Trans. Syst. Man Cybern. 45 (9) (2015) 1245–1253.
[29] S. Kabbur, X. Ning, G. Karypis, FISM: factored item similarity models for top-n recommender systems[c], in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2013, pp. 659–667.
[30] X. Chen, W. Pan, Z. Ming, TOCCF: Time-aware one-class collaborative filtering[c], The Workshop on Multi-Dimensional Information Fusion for User Modeling and Personalization, 2016.
[31] D.M. Blei, A.Y. Ng, M.I. Jordan, Latent dirichlet allocation[j], J. Mach. Learn. Res. 3 (2003) 993–1022.
[32] Y. Chen, X. Yin, Z. Li, X. Hu, J.X. Huang, A LDA-based approach to promoting ranking diversity for genomics information retrieval[j], BMC Genomics 13 (Suppl 3) (2012) S2.
[33] J. Wilson, S. Chaudhury, B. Lall, Improving collaborative filtering based recommenders using topic modelling[c], in: Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)-Volume 01. IEEE Computer Society, 2014, pp. 340–346.
[34] T.M. Chang, W.F. Hsiao, LDA-based personalized document recommendation[c], in: Pacific Asia Conference on Information Systems, 2013.
[35] T. Hofmann, Probabilistic latent semantic indexing[c], in: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 1999, pp. 50–57.
[36] D. Newman, P. Smyth, M. Welling, A.U. Asuncion, Distributed inference for latent dirichlet allocation[c], Adv. Neural Inf. Process Syst. (2008) 1081–1088.
[37] Y. Wang, H. Bai, M. Stanton, W.Y. Chen, E.Y. Chang, PLDA: Parallel latent dirichlet allocation for large-scale applications[c], in: International Conference on Algorithmic Applications in Management, 2009, pp. 301–314.
[38] G. Heinrich, Parameter Estimation for Text Analysis[r], Technical report, 2005.
[39] Z. Gantner, S. Rendle, C. Freudenthaler, L. Schmidt-Thieme, Mymedialite: a free recommender system library[c], in: Proceedings of the Fourth ACM Conference on Recommender Systems(RecSys), 2011, pp. 305–308.

Please cite this article as: H. Zhang et al., Incorporating temporal dynamics into LDA for one-class collaborative filtering, Knowledge-Based Systems (2018), https://doi.org/10.1016/j.knosys.2018.02.036