Frontiers of Information Technology & Electronic Engineering
www.zju.edu.cn/jzus;engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Personalized topic modeling for recommending user-generated content*

Wei ZHANG[1,2], Jia-yu ZHUANG[3,4], Xi YONG[1,2,5], Jian-kou LI[1,2], Wei CHEN[3,4], Zhe-min LI[†‡3,4]

([1]*State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China*)

([2]*School of Information Science and Engineering, University of Chinese Academy of Sciences, Beijing 100190, China*)

([3]*Agricultural Information Institute, Chinese Academy of Agricultural Sciences, Beijing 100081, China*)

([4]*Key Laboratory of Agri-information Service Technology, Ministry of Agriculture, Beijing 100081, China*)

([5]*Water Information Centre, Ministry of Water Resources, Beijing 100053, China*)

[†]E-mail: lizhemin@caas.cn

Received Nov. 17, 2015; Revision accepted Apr. 25, 2016; Crosschecked Apr. 28, 2017

Abstract: User-generated content (UGC) such as blogs and twitters are exploding in modern Internet services. In such systems, recommender systems are needed to help people filter vast amount of UGC generated by other users. However, traditional recommendation models do not use user authorship of items. In this paper, we show that with this additional information, we can significantly improve the performance of recommendations. A generative model that combines hierarchical topic modeling and matrix factorization is proposed. Empirical results show that our model outperforms other state-of-the-art models, and can provide interpretable topic structures for users and items. Furthermore, since user interests can be inferred from their productions, recommendations can be made for users that do not have any ratings to solve the cold-start problem.

Key words: User-generated content (UGC); Collaborative filtering (CF); Matrix factorization (MF); Hierarchical topic modeling
http://dx.doi.org/10.1631/FITEE.1500402 **CLC number:** TP393.0

1 Introduction

User-generated content (UGC) refers to any form of content including blogs, images, or videos that are generated by the users of a website. It is an important concept of Web 2.0, and has become increasingly popular in modern Internet services such as blog websites (e.g., Blogger, Tumblr, and Word Press), social networking sites (e.g., Facebook, Twitter, and Instagram), and video sharing sites (e.g., YouTube). On these websites, users can create and publish their own productions, and also review or rate items generated by others. With the explosive growth of UGC, recommender systems are needed to help users filter vast amounts of content generated by others.

Two main techniques for recommender systems are collaborative filtering (CF) (Linden *et al.*, 2003; Deshpande and Karypis, 2004; Salakhutdinov and Mnih, 2007) and content-based methods (Mooney and Roy, 2000; Melville *et al.*, 2002; Lops *et al.*, 2011). CF-based approaches automatically extract user interests from their rating history on items. The intuition of CF is that users with similar rating behavior in their history are assumed to have similar tastes, and will have similar ratings in the future. A key advantage of CF is that it does not rely on the specific content of items; therefore, it can recommend any kind of item that has been rated. However, this is also the weakness of CF since it brings data sparsity and cold-start problems. As a complement,

‡ Corresponding author

* Project supported by the Monitoring Statistics Project on Agricultural and Rural Resources, MOA, China, the Innovative Talents Project, MOA, China, and the Science and Technology Innovation Project Fund of Chinese Academy of Agricultural Sciences (No. CAAS-ASTIP-2015-AII-02)

ORCID: Zhe-min LI, http://orcid.org/0000-0003-3170-0117

content-based methods use content-based features, and recommend items that are similar to what a user has liked in the past. More complex models that combine CF and content-based methods have also been proposed (Agarwal and Chen, 2009; 2010).

These traditional approaches also apply to the recommendation of UGC. However, the key information of UGC, authorship, is ignored, and it is crucial for recommendation tasks since user productions indicate directly their preferences. In this study, we aim to develop a generative model that considers user authorship of items for UGC-based systems, and we will focus on discrete items such as blogs and articles. To describe the authorship of UGC, we use terms user/author and item/production interchangeably in this paper, depending on the context.

Before going into the details, we list several considerations for building such a model. In UGC-based systems, a user's preferences are reflected in two aspects. The first aspect is the user's productions, and the second is his/her rating behavior on items generated by other users. Apparently, for each user, these two aspects are related closely to each other. On one hand, they should be similar—a researcher publishing papers mainly in the data mining domain would like articles in the same domain. On the other hand, they may be different—the same researcher may also like articles in the mathematical or physical domain. Likewise, characteristics of an item are reflected in two aspects: its content and its ratings given by users. For instance, if we find that an article is about mathematics in terms of its content, then we can predict that it should be preferred mainly by mathematicians. However, from the ratings of that article, we may further identify that it is also popular for data mining researchers, while this information may be difficult to learn purely from the content.

Taking the above into consideration, we propose a probabilistic generative model that combines topic modeling and matrix factorization (MF). We use hierarchical topic modeling to model item content and authorship information, and use MF to model user's rating behavior. The relationship between these two models is described with regression models (Agarwal and Chen, 2009; Wang and Blei, 2011), and is balanced automatically according to the data. If a user has rated only a few items, recommendations will be based mainly on the user's productions; otherwise, they will be based on his/her ratings. Similarly, an item with few ratings will be recommended based mainly on its content; otherwise, it will be recommended based mainly on its ratings. Compared to other content-based models, our approach can recommend new items to users. Furthermore, by incorporating authorship, we can infer user's preferences from their productions, and hence can solve the cold-start problem to make recommendations for users who have no ratings.

We conducted experiments on scientific articles in the data mining domain, collected from the Microsoft Academy Search Engine. Empirical results showed that our model outperforms other state-of-the-art ones, hence revealing the importance of authorship for recommendation tasks for UGC. We also studied the capability of our model for cold-start recommendations, and have obtained better results compared with others. The main contributions of our paper are listed as follows:

1. We propose a unified generative model for recommending UGC.

2. We combine hierarchical topical modeling with CF.

3. Our model can provide interpretable topic representations for both users and items (Section 4.6).

2 Related work

Collaborative filtering (CF) based (Linden *et al.*, 2003; Deshpande and Karypis, 2004) and content-based methods are two main techniques for recommender systems. CF-based approaches can be grouped into two general classes, namely neighborhood- and model-based methods. Among model-based approaches, matrix factorization (MF) has recently received much interest and achieved much progress (Salakhutdinov and Mnih, 2007; Salakhutdinov and Mnih, 2008). Several extensions of MF have been proposed to handle implicit feedbacks (Hu *et al.*, 2008; Pan *et al.*, 2008; Rendle *et al.*, 2009).

In contrast with CF-based approaches, which are based purely on ratings, content-based methods (Mooney and Roy, 2000; Melville *et al.*, 2002; Agarwal and Chen, 2009) make recommendations by additionally analyzing the items' content. For items in the form of text documents, topic models such as

latent Dirichlet allocation (LDA) (Blei *et al.*, 2003) are introduced to build more complex models (Agarwal and Chen, 2010; Wang and Blei, 2011; Purushotham *et al.*, 2012; Li *et al.*, 2013). Recommendations with UGC are also considered (de Pessemier *et al.*, 2011; Xu and Yin, 2015; Xu *et al.*, 2015), and various types of UGC are used to enhance the recommendation accuracy on other target items. In this study, we consider a different problem of recommending UGCs themselves, which means that target items are UGCs generated by other users.

3 Proposed model

In this section, we describe our model: personalized topic regression (PTR). PTR combines hierarchical topic modeling with MF. We first introduce hierarchical Dirichlet allocation (HDA) which serves as a building block of our model. Then we briefly describe collaborative topic regression (CTR), the baseline model of this study. Finally we present PTR in detail and develop an efficient algorithm for learning parameters of the model.

The following notations are used throughout the paper:

$\boldsymbol{\theta}_i$: topic proportions of user i,
$\boldsymbol{u}_i$: latent factors of user i,
$\boldsymbol{\varphi}_j$: topic proportions of item j,
$\boldsymbol{v}_j$: latent factors of item j,
z_{jn}: topic assignment of the nth word of item j,
ω_{jn}: the nth word of item j,
$r_{i,j}$: the rating of user i for item j.

3.1 Hierarchical Dirichlet allocation

LDA aims to discover topic proportions of documents from one corpus. These topic proportions provide low-dimensional representations of the documents. In this subsection, we consider the problem of modeling documents in multiple corpora. In this case, documents are organized into a two-level hierarchical structure. Hence, hierarchical models such as hierarchical Dirichlet process (HDP) (Blei *et al.*, 2010; Teh *et al.*, 2004) are proposed to address this problem. For our modeling purpose, we use HDA (Veeramachaneni *et al.*, 2005), which can be considered as the parametric version of HDP. Given topic parameters $\boldsymbol{\beta}=[\beta_1, \beta_2, \cdots, \beta_k]$, the generative process of HDA is presented in Algorithm 1, in which Mult(·) represents the multinomial distribution and Dir(·) the Dirichlet distribution.

Algorithm 1 Generative process of hierarchical Dirichlet allocation

For each corpus i
  Draw corpus-level topic proportions $\boldsymbol{\theta}_i \sim \mathrm{Dir}(\alpha)$
End for
For each document j
  Draw document-level topic proportions $\boldsymbol{\theta}_j \sim \mathrm{Dir}(\sigma\boldsymbol{\theta}_{\alpha(j)})$
  // $\alpha(j)$ is the corpus to which document j belongs
  For each word ω_{jn} in document j
    Draw topic assignment $z_{jn} \sim \mathrm{Mult}(\boldsymbol{\varphi}_j)$
    Draw word $\omega_{jn} \sim \mathrm{Mult}(\boldsymbol{\varphi}_{z_{jn}})$
  End for
End for

In this model, both documents and corpora are projected into a low-dimensional topic space. Document-level topic proportions $\boldsymbol{\varphi}_j$ represent the theme of that document, while corpus-level topic proportions $\boldsymbol{\theta}_i$ represent the theme of the whole corpus. Topic proportions of a document j are assumed to be similar to topic proportions of corpus $\alpha(j)$ to which it belongs, and we model this idea by assuming that $\boldsymbol{\varphi}_j$ comes from a Dirichlet distribution with means $\boldsymbol{\theta}_{\alpha(j)}$ and precision σ. The precision parameter σ controls how much the distributions diverge from its mean. Thus, a high σ means that documents are assumed to have similar topic proportions to their corpus-level topic proportions.

3.2 Collaborative topic regression

Our model is closely related to CTR (Wang and Blei, 2011). In fact, CTR can be considered as a special case of our model when all items come from the same default user. Given topic proportion parameters $\boldsymbol{\beta}$, the generative process of CTR is defined in Algorithm 2.

Fig. 1 shows the graphical representation of CTR. CTR uses LDA to learn topic proportions of items and MF to model user ratings. The latent factors of an item are generated from a Gaussian distribution. The mean of the Gaussian is set to be the topic proportions of the item. This reveals the idea that the recommendation of an item is based on its content. Moreover, as more users rate items, we can make better recommendations by analyzing the ratings. CTR does a good job in using item content for recommendations.

However, it does not consider the authorship information of UGC. We can significantly improve the recommendation performance of UGC-based systems by additionally modeling authorship, which will be demonstrated later.

Algorithm 2 Generative process of collaborative topic regression

For each user i
  Draw user latent factors $\boldsymbol{u}_i \sim N(\boldsymbol{0}, \lambda_u^{-1} \cdot \boldsymbol{I})$
End for
For each item j
  Draw item topic proportions $\boldsymbol{\varphi}_j \sim \text{Dir}(\alpha)$
  Draw item latent factor $\boldsymbol{v}_j \sim N(\boldsymbol{0}, \lambda_v^{-1} \cdot \boldsymbol{I})$
  // $N(\cdot, \cdot)$ represents the Gaussian distribution
  For each word ω_{jn}
    Draw topic assignment $z_{jn} \sim \text{Mult}(\boldsymbol{\varphi}_j)$
    Draw word $\omega_{jn} \sim \text{Mult}(\boldsymbol{\varphi}_{z_{jn}})$
  End for
End for
For each rating pair (i, j)
  Draw rating $r_{i,j} \sim N(\boldsymbol{u}_i^{\mathrm{T}} \boldsymbol{v}_j, c_{i,j}^{-1})$
End for

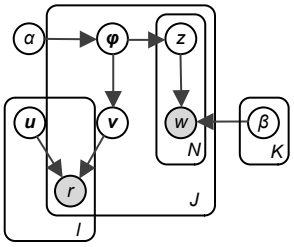

Fig. 1 Graphical representation of collaborative topic regression (λ_u and λ_v are omitted for brevity)

3.3 Personalized topic regression

First, let us consider modeling item content and user authorship. By incorporating authorship, items are naturally organized into a two-layer hierarchical structure—each user has a personal corpus, and each corpus contains items produced by the corresponding user. Hence, we can use hierarchical topic modeling to model authorship, and project each user and each item into a latent topic space. In that space, each user i is represented by a topic proportions vector $\boldsymbol{\theta}_i$ that indicates i's preferred topics. This is the reason why we call this model personalized topic modeling. Likewise, each item j is represented by a topic proportions vector $\boldsymbol{\varphi}_j$ that characterizes j's content. The hierarchical structure expresses that, when producing a specific item, user i will choose a topic for that item proportions vector $\boldsymbol{\varphi}_j$ that is close to $\boldsymbol{\theta}_i$.

Second, we model user's rating behavior via MF. In MF, each user and each item are projected into a latent low-dimensional space—user i is represented by latent factors $\boldsymbol{u}_i$ that indicate his/her interest on other items, and item j is represented by latent factors $\boldsymbol{v}_j$ that represent its latent features.

To model the data completely, we need to specify the relationship between the above two models. With these models, each user i has two representations: the topic proportions vector $\boldsymbol{\theta}_i$, and the latent factors vector $\boldsymbol{u}_i$. Apparently, these two vectors are closely correlated—if a user always publishes articles on a certain topic, the user will be likely to prefer other articles on the same topic. Like Wang and Blei (2011), we model this relationship with regression models:

$$\boldsymbol{u}_i \sim N(\boldsymbol{\theta}_i, \lambda_u^{-1} \cdot \boldsymbol{I}), \tag{1}$$

where λ_u is the precision parameter. This is why we call this model personalized topic regression (PTR). Similarly, for each item, we model the relationship between $\boldsymbol{\varphi}_j$ and $\boldsymbol{v}_j$ as follows:

$$\boldsymbol{v}_j \sim N(\boldsymbol{\varphi}_j, \lambda_v^{-1} \cdot \boldsymbol{I}), \tag{2}$$

where λ_v is the precision parameter. Based on the above consideration, we propose the generative process of PTR in Algorithm 3, in which we assume there are k topics $\boldsymbol{\beta}=[\beta_1, \beta_2, \cdots, \beta_k]$.

Algorithm 3 Generative process of personalized topic regression

For each user i
  Draw user topic proportions $\boldsymbol{\theta}_i \sim \text{Dir}(\alpha)$
  Draw user latent factor $\boldsymbol{u}_i \sim N(\boldsymbol{\theta}_i, \lambda_u^{-1} \cdot \boldsymbol{I})$
End for
For each item j
  Draw item topic proportions $\boldsymbol{\varphi}_j \sim \text{Dir}(\alpha \boldsymbol{\theta}_{\alpha(j)})$
  // $\alpha(j)$ is the user (corpus) to which item j belongs
  Draw item latent factor $\boldsymbol{v}_j \sim N(\boldsymbol{\varphi}_j, \lambda_v^{-1} \cdot \boldsymbol{I})$
  For each word ω_{jn}
    Draw topic assignment $z_{jn} \sim \text{Mult}(\boldsymbol{\varphi}_j)$
    Draw word $\omega_{jn} \sim \text{Mult}(\boldsymbol{\varphi}_{z_{jn}})$
  End for
For each rating pair (i, j)
  Draw rating $r_{i,j} \sim N(\boldsymbol{u}_i^{\mathrm{T}} \boldsymbol{v}_j, c_{i,j}^{-1})$
End for

Fig. 2 shows the graphical representation of PTR. We allow user i's latent factor $\boldsymbol{u}_i$ to deviate from user i's topic proportions $\boldsymbol{\theta}_i$. If user i shows wider interest from rating behavior than from productions, the deviation between $\boldsymbol{u}_i$ and $\boldsymbol{\theta}_i$ should be large; otherwise, the deviation should be small. Likewise, if an item j is liked by users from diverse domains, the deviation between $\boldsymbol{v}_i$ and $\boldsymbol{\varphi}_j$ should also be large. Precision parameters λ_u and λ_v control the allowed deviation between user's productions and their interested items. σ controls how far the topic of user i's production can deviate from user i's personal topic proportions. A small σ means that users have a wide range of interest and are inclined to generate diverse items, while a large σ means that users focus their interest on one theme and tend to generate similar items. c_{ij} indicates our confidence on the observed ratings $r_{i,j}$. The larger the value c_{ij}, the more we trust the rating $r_{i,j}$. This is useful for modeling datasets with implicit feedbacks. In this case, $r_{i,j}$=0 has two interpretations—user i may not like item j or may just be unaware of it. Hence, we set c_{ij} to be a higher value when $r_{i,j}$=1, and a lower value when $r_{i,j}$=0:

$$c_{ij}=\begin{cases}a, & r_{i,j}=1,\\ b, & r_{i,j}=0,\end{cases} \tag{3}$$

where $a>b>0$ are parameters to be tuned according to the data.

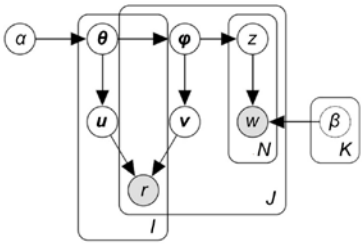

Fig. 2 Graphical representation of PTR (λ_u, λ_v, and σ are omitted for brevity)

Links between $\boldsymbol{\theta}_i$ and $\boldsymbol{\varphi}_j$ are connected according to the data—if user i is item j's author, a link between $\boldsymbol{\theta}_i$ and $\boldsymbol{\varphi}_j$ is added

There are cases in which items (e.g., Wikis, scientific articles) have multiple authors. PTR can be easily extended to model these datasets. Recall that in the one author case, the mean for an item is its author's topic proportions. If an item has multiple authors, we set its mean to be the average topic proportions of all its authors:

$$\boldsymbol{\varphi}_j \sim \mathrm{Dir}(\sigma\overline{\boldsymbol{\theta}}_{i\in A(j)}), \tag{4}$$

where $A(j)$ denotes the author set of item j.

3.4 Learning the parameters

Given λ_u, λ_v, σ, and topic parameters $\boldsymbol{\varphi}_j$ and $\boldsymbol{\beta}$, learning the full posterior $\boldsymbol{u}_i$, $\boldsymbol{v}_j$, $\boldsymbol{\theta}_i$, and $\boldsymbol{\varphi}_j$ is analytically intractable. As in Wang and Blei (2011), we develop an EM-style algorithm that learns the maximum a posteriori (MAP) estimates.

Let $p(i)$ denote the set of productions that belong to user i. Maximizing the posterior is equivalent to maximizing the complete log-likelihood:

$$\begin{aligned}L=&\sum_i\sum_{j\in p(i)}\sum_k[\log\Gamma(\sigma\theta_{ik})+(\sigma\theta_{ik}-\alpha\log\varphi_{jk})]\\&-\frac{\lambda_u}{2}\sum_i(\boldsymbol{u}_i-\boldsymbol{\theta}_i)^{\mathrm{T}}(\boldsymbol{u}_i-\boldsymbol{\theta}_i)\\&-\frac{\lambda_v}{2}\sum_j(\boldsymbol{v}_j-\boldsymbol{\varphi}_j)^{\mathrm{T}}(\boldsymbol{v}_j-\boldsymbol{\varphi}_j)\\&+\sum_j\sum_n\log\sum_k\varphi_{jk}\beta_{k,\omega_{jn}}-\sum_{i,j}\frac{c_{ij}}{2}(r_{i,j}-\boldsymbol{u}_i^{\mathrm{T}}\boldsymbol{v}_j)^2,\end{aligned} \tag{5}$$

where we have omitted $\sum_k\theta_{ik}$=1 and other constants, and set α=1.

To optimize $\boldsymbol{U}$=[$\boldsymbol{u}_1$, $\boldsymbol{u}_2$, …, $\boldsymbol{u}_I$] and $\boldsymbol{V}$=[$\boldsymbol{v}_1$, $\boldsymbol{v}_2$, …, $\boldsymbol{v}_J$], we compute the gradient with respect to $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$, respectively, and set them to zero. Then we obtain the following updating equations:

$$\boldsymbol{u}_i=(\boldsymbol{V}\boldsymbol{C}_i\boldsymbol{V}^{\mathrm{T}}+\lambda_u\boldsymbol{I}_K)^{-1}(\boldsymbol{V}\boldsymbol{C}_i\boldsymbol{R}_i+\lambda_u\boldsymbol{\theta}_i), \tag{6}$$

$$\boldsymbol{v}_j=(\boldsymbol{U}\boldsymbol{C}_j\boldsymbol{U}^{\mathrm{T}}+\lambda_v\boldsymbol{I}_K)^{-1}(\boldsymbol{U}\boldsymbol{C}_j\boldsymbol{R}_j+\lambda_v\boldsymbol{\varphi}_j), \tag{7}$$

where $\boldsymbol{C}_i$ is a diagonal matrix and $\boldsymbol{R}_i=(r_{i,j})_{j=1}^{J}$. We can significantly speed up this algorithm in the case of implicit feedbacks. For further details, the readers are referred to Hu *et al.* (2008).

To optimize user topic proportions $\boldsymbol{\theta}_i$, we separate items containing $\boldsymbol{\theta}_i$ and obtain the following optimization problem:

$$\begin{aligned}\max_{\boldsymbol{\theta}_i}\ &\sum_{j\in p(i)}(-\log\Gamma(\sigma\theta_{ik})+\sigma\theta_{ik}\log\varphi_{jk})\\&-\frac{\lambda_u}{2}\sum_i(\boldsymbol{u}_i-\boldsymbol{\theta}_i)^{\mathrm{T}}(\boldsymbol{u}_i-\boldsymbol{\theta}_i)\\\text{s.t.}\ &\sum_k\theta_{ik}=1.\end{aligned} \tag{8}$$

We cannot solve this problem analytically, so we use the projection gradient (Duchi *et al.*, 2008). To optimize item topic proportions $\boldsymbol{\varphi}_j$, we first define $q(z_{jn}=k)=\gamma_{jnk}$. Then we separate items containing $\boldsymbol{\varphi}_j$ and apply Jensen's inequality:

$$\begin{aligned}\mathrm{L}(\boldsymbol{\varphi}_j) \geq & \sum_k (\sigma\theta_{\alpha(j),k}-1)\log\varphi_{jk} \\ & -\frac{\lambda_v}{2}\sum_j (\boldsymbol{v}_j-\boldsymbol{\varphi}_j)^{\mathrm{T}}(\boldsymbol{v}_j-\boldsymbol{\varphi}_j) \\ & +\sum_n\sum_k \gamma_{jnk}[\log(\varphi_{jk}\beta_{k,\omega_{jn}})-\log\gamma_{jnk}] \\ = & \mathrm{L}(\boldsymbol{\varphi}_j,\boldsymbol{\gamma}_j),\end{aligned} \tag{9}$$

where $\alpha(j)$ is the user (corpus) to which item j belongs. To make inequality (9) tight, the optimal γ_{jnk} satisfies $\gamma_{jnk}\propto\varphi_{jk}\beta_{k,\omega_{jn}}$. We cannot optimize $\boldsymbol{\varphi}_j$ analytically, and we use the projection gradient again.

After we estimate $\boldsymbol{U}$, $\boldsymbol{V}$, $\boldsymbol{\theta}$, and $\boldsymbol{\varphi}$, we can optimize $\boldsymbol{\beta}$ as

$$\beta_{k\omega}\propto\sum_j\sum_n \gamma_{jnk}1[\omega_{jn}=\omega], \tag{10}$$

where $1[\cdot]$ is the Boolean function. The complete process is described in Algorithm 4.

Algorithm 4 Learning algorithm of our personalized topic regression model

Input: ratings and authorship between users and items
Output: model parameters $\boldsymbol{u}_i$, $\boldsymbol{v}_j$, $\boldsymbol{\theta}_i$, $\boldsymbol{\varphi}_j$, and $\boldsymbol{\beta}$
Initialize parameters $\boldsymbol{u}_i$, $\boldsymbol{v}_j$, $\boldsymbol{\theta}_i$, $\boldsymbol{\varphi}_j$, and $\boldsymbol{\beta}$
Repeat:
  For each rating c_{ij}
    Update $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ using Eqs. (6) and (7)
    Update $\boldsymbol{\theta}_i$ and $\boldsymbol{\varphi}_j$ using problems (8) and (9)
    Update $\boldsymbol{\beta}$ using Eq. (10)
  End for
Until convergence

3.5 Prediction

We refer to users that have at least one rating as active users and other users as inactive users. Similarly, we refer to items that have received at least one rating as active items and other items as inactive items.

We consider three types of recommendation tasks. The first one is recommending active items to active users, which we call warm-start recommendation. In general, the prediction is estimated as follows:

$$\boldsymbol{E}[r_{i,j}\mid D]\approx\boldsymbol{E}[\boldsymbol{u}_i\mid D]^{\mathrm{T}}\boldsymbol{E}[\boldsymbol{v}_i\mid D]. \tag{11}$$

We approximate these expectations using the point estimations of $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$:

$$r^*_{i,j}=(u^*_{ij})^{\mathrm{T}}v^*_j. \tag{12}$$

We then rank the predicted ratings to make a recommendation list for each user.

The second task is item cold-start recommendations, which refer to recommending inactive items to active users. In this case, we do not have rating information about the items to be recommended. Therefore, we approximate ratings on these items based on their content:

$$r^*_{i,j}=(u^*_{ij})^{\mathrm{T}}\varphi^*_j. \tag{13}$$

The third task is user cold-start recommendations, which refer to recommending active items to inactive users. As in the previous case, we approximate the ratings as follows:

$$r^*_{ij}=(\theta^*_{ij})^{\mathrm{T}}v^*_j. \tag{14}$$

4 Empirical results

4.1 Dataset

The data used are scientific articles from a data mining domain collected by the Microsoft Academy Search Engine. For each article, we have its title, abstract, authors, and references. We treat authors as users and articles as items. If an author u_i cites an article v_j at least once, we consider the rating r_{ij} as 1; otherwise, we consider r_{ij} as 0.

We remove articles with no abstracts, and remove users with fewer than 10 ratings to obtain a dataset containing 10 390 authors, 38 733 articles, 284 144 rating pairs, and 74 774 authorship pairs. On average, each user has 27 ratings and about 4 productions.

For each article, we concatenate its title and

abstract. After removing stop words, we use tf-idf to choose the top 8000 distinct words as the vocabulary. Finally, we obtain a corpus of 3.1 million words. We split the dataset into two parts: 80% of the data is used for training and the rest for testing.

4.2 Evaluation metrics

In recommender systems, users are usually presented with a ranked list of items that might be of interest. Hence, we focus on ranking performance in our experiments. Precision and recall are two alternative metrics. However, in our experiments, precision cannot be accurately computed, because zero ratings are uncertain—a user may not like the item or even does not know about it. Hence, we choose recall as our evaluation metric. Suppose we present each user with M items. The definition of recall@M for that user is as follows:

$$\text{recall@}M=\frac{\text{Number of items the user likes in top-}M\text{ list}}{\text{Total number of items the user likes}}. \quad (15)$$

4.3 Comparisons

In this section, we compare our model (PTR) with CTR and MF for warm-start and cold-start recommendation tasks.

4.3.1 Warm-start recommendations

First, we study the performance for warm-start recommendations. All of the three models can make warm-start recommendations. We use cross validation to find parameters for a good performance. For all experiments of the three models, we set K=200, a=1, and b=0.01. For MF, we set $\lambda_u=\lambda_v$=0.01. For CTR, we set λ_u=0.01 and λ_v=100. For PTR, we set $\lambda_u=\lambda_v$=1. From these parameter settings, we can identify different characteristics of the three models. Recall that precision parameters λ_u and λ_v express our confidence on the similarity between the topic proportions and latent factors for a given object (user or item). Larger λ_u means that we are more confident that user i's latent factors vector $\boldsymbol{\theta}_i$ is close to user i's topic proportions $\boldsymbol{\theta}_i$. In MF, we do not have content features, and thus we set our confidences λ_u and λ_v to be a small value 0.01. In CTR, as we have content features on the item side, we set a higher value for λ_v to relatively fix item j's latent factors $\boldsymbol{v}_j$ to its content feature $\boldsymbol{\varphi}_j$, and set a small value for λ_u to relax $\boldsymbol{u}_i$ and let it be learned from the rating behavior. In PTR, we have content features on both users and items, and hence we choose a medium value to balance constraints from both sides. As will be shown later, PTR is insensitive to σ for recommendation tasks. Therefore, we fix σ=1 for all comparisons.

Fig. 3 shows that PTR outperforms CTR and MF in terms of the recall metric as we vary the number of recommended items. Specifically, the advantage of PTR is more embodied when M is small, in which case PTR significantly improves the performance compared to CTR and MF. This demonstrates that, with authorship information, PTR can better identify user's personal interest and make more precise recommendations on top results. This is essential for recommender systems, since users usually focus only on top recommendations. When M is large, sparse ratings are insufficient for MF to make good recommendations; content features become more important and help PTR and CTR achieve better performance.

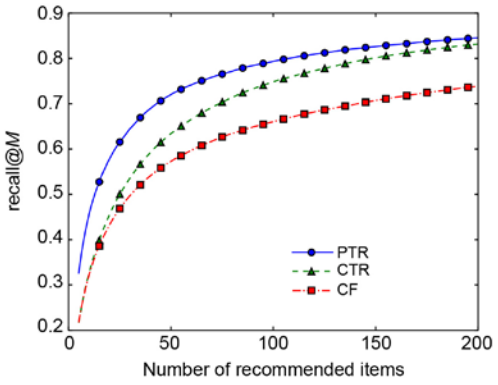

Fig. 3 Recall comparison for warm-start recommendations by varying the number of recommended items

For all models, we set a=1 and b=0.01. For MF, we set $\lambda_u=\lambda_v$=0.01; for CTR, we set λ_u=0.01 and λ_v=100; for PTR, we set $\lambda_u=\lambda_v=\sigma$=0.01

4.3.2 Cold-start recommendations

We compare the performance for cold-start recommendation tasks. We consider two types of cold-start problems: item cold-start recommendations and user cold-start recommendations. Note that MF cannot do cold-start recommendations, and CTR can do only item cold-start recommendations. Therefore, we compare our model with CTR only on item cold-start recommendation tasks.

For item cold-start recommendations, we use a 5-fold cross validation. We randomly group all items into five folds. For items in each fold, we pick out all rating pairs on them to build a test set, and leave the remaining rating pairs as a training set. We then iteratively fit the models to each training set, and evaluate the performance on the corresponding test set (we make recommendations only on cold-start items).

Our aim is to learn good user representations based on items' content. Thus, we choose a high λ_v to tie together items' latent factors $\boldsymbol{v}_j$ with their topic proportions $\boldsymbol{\varphi}_j$, and let the latent user factors $\boldsymbol{u}_i$ be flexible. For CTR, we also set λ_u=0.01 and λ_v=100. For PTR, we set λ_u=1 and λ_v=100.

For user cold-start recommendations, we group users into five folds and process the data similarly as in the previous case. Note that in this case, we make recommendations for each cold-start user on all items. As shown in Fig. 4, this is a hard problem and the recall is low.

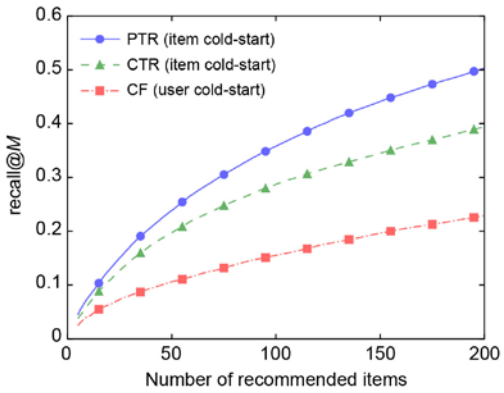

Fig. 4 Recall comparison for cold-start recommendations by varying the number of recommended items

For all models, we set a=1 and b=0.01. Both CTR and PTR can perform item cold-start recommendations. For CTR, we set λ_u=0.01 and λ_v=100. For PTR, we set λ_u=1, λ_v=100, and σ=1. Only PTR can do user cold-start recommendations, and we set λ_u=100 and λ_v=1

4.4 Impact of σ

The main difference between our model and CTR is the HDA used for modeling user authorship. In this section, we study the effect of the precision parameter σ in HDA.

For each user i, the precision parameter σ controls how much the topic proportions of user i's productions diverge from his/her personal topic proportions $\boldsymbol{\theta}_i$. We use $\overline{\boldsymbol{\varphi}(p(i))}$ to denote the average topic proportions of all i's productions. When σ is large, we expect that $\overline{\boldsymbol{\varphi}(p(i))}$ is close to $\boldsymbol{\theta}_i$. Since both vectors are probability measures, we use KL-divergence to measure the difference between these two vectors. As shown in Fig. 5a, with the increase of σ, the average KL-divergence decreases and reaches almost zero when σ=1.

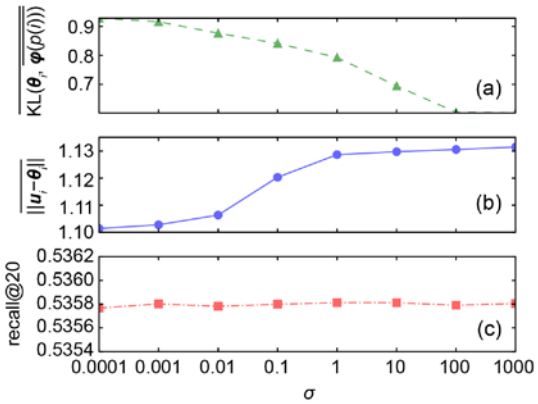

Fig. 5 Impact of σ: (a) average KL-divergence with the increment of σ; (b) average Euclidean distance between $\boldsymbol{u}_i$ and $\boldsymbol{\theta}_i$; (c) the performance of recommendations

$\overline{\boldsymbol{\varphi}(p(i))}$ denotes the average topic proportions of user i's productions, KL $(\cdot)$ the KL divergence, and $\|\cdot\|$ the Euclidean distance

When σ is large, even though user i's topic proportions are tied with their productions, the latent factors $\boldsymbol{u}_i$ are still flexible enough to compensate for the loss. This is confirmed in Fig. 5b, in which the average Euclidean distance between $\boldsymbol{u}_i$ and $\boldsymbol{\theta}_i$ increases as σ becomes larger. The main task of our model is to make recommendations. As shown in Fig. 5c, the performance of recommendations fluctuates only slightly as σ varies. The reason is that users' latent factors $\boldsymbol{\theta}_i$ diverge automatically from their topic proportions to balance the effect of changing σ.

Nonetheless, for recommendation tasks, our model is insensitive to σ. Hence, we set σ=1 in all our experiments.

4.5 Relationship with object properties

In this section, we study how the performance and parameters of our model vary as functions of object properties: the number of productions or ratings for users, and the number of authors or ratings for items.

We first study the relationship between parameters and user properties. In PTR, each user i has two

representations: topic proportions $\boldsymbol{\theta}_i$ and latent factors $\boldsymbol{u}_i$. We are interested in the distances between the two representations, as the distances measure the differences between user's production features and their rating behavior. As shown Fig. 6a, the distances tend to have a larger variance for users with fewer productions. This is reasonable, since we cannot accurately infer user's topic proportions when they have only a few productions. Fig. 6a also shows that the distances tend to decrease as users publish more articles. This is because a large number of productions ensure a better identification of a user's preferences; hence, the latent factors $\boldsymbol{u}_i$ do not need to diverge from the topic proportions $\boldsymbol{\theta}_i$ to learn more preferences from the ratings.

Fig. 6b shows that the distances tend to increase for users with more ratings. The reason is that, when rating more items, a user tends to exhibit more diverse preferences that cannot be found in the user's productions.

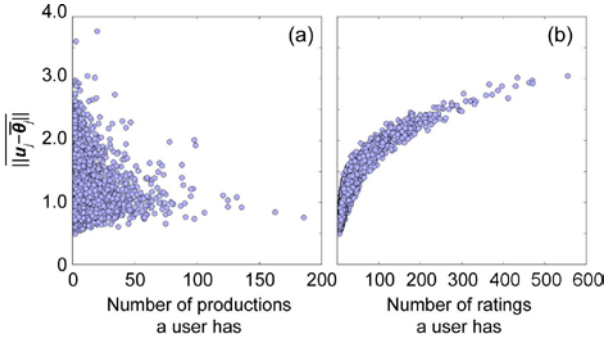

Fig. 6 Distances between users' two representations θ_i and u_i vary as functions of the number of users' productions (a) or ratings (b)

Likewise, we inspect items' properties in a similar way. Fig. 7 shows how the distances between items' two representations $\boldsymbol{\varphi}_j$ and $\boldsymbol{v}_j$ vary as functions of the number of items' authors or received ratings. From Fig. 7a, we observe that the distances tend to decrease as items have more authors. The reason is that, multiple authors make the topic proportions $\boldsymbol{\varphi}_j$ of an item j diverse enough to capture the item's latent features, and hence it leaves only a little work for the latent factors $\boldsymbol{v}_j$ to do.

Fig. 7b shows that the distances increase as items receive more ratings. This is because items with more ratings tend to exhibit more topics that cannot be learned purely from their content.

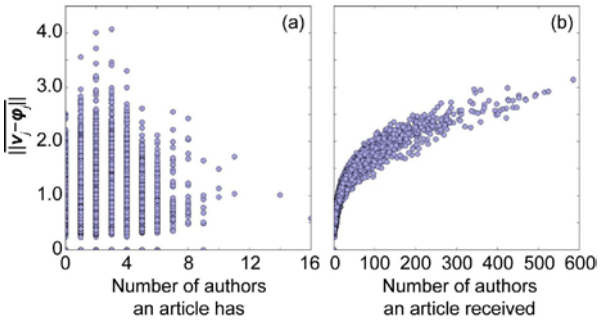

Fig. 7 Distances between items' two representations φ_j and v_j vary as functions of the number of j's authors (a) or received ratings (b)

Next, we study how the recall performance varies as functions of user properties. From Fig. 8a, we observe an upward trend as users have more productions. This indicates that we can better identify the preferences of a prolific user. In contrast, we observe a downward trend from Fig. 8b. The reason is that users with more ratings tend to rate more infrequent items that are hard to recommend. In both cases, users with few productions or few ratings tend to have large variances in their predictions.

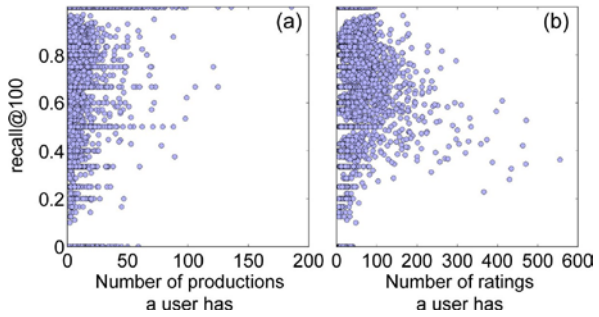

Fig. 8 Recall varies as a function of the number of users' productions (a) or ratings (b)

4.6 Examining topic spaces

In this subsection, we examine the topic spaces learned from our model and CTR. CTR does not explicitly model users' topic proportions. However, in CTR, we can explain users' latent factors based on topics learned from the item side. For each user i, we can find i's top preferred topics by ranking the entries of i's latent factors vector $\boldsymbol{u}_i$.

The topic preferences learned by CTR are based purely on user ratings. They indicate what types of items a user would like to give positive ratings. In PTR, as we have users' authorship information, we explicitly model each user i's topic proportions $\boldsymbol{\theta}_i$

based on their productions. By ranking the entries of $\boldsymbol{\theta}_i$, we can further learn what types of items user i would like to create.

In PTR, $\boldsymbol{\theta}_i$ serves as the prior of user i's latent factors $\boldsymbol{u}_i$, which express the intuition that user i should like items similar to his/her own productions. However, user i may also like other types of items. These further preferences can be learned from user i's rating history and embodied in his/her latent factors $\boldsymbol{u}_i$. To distinguish from latent factors learned by CTR, we denote the vectors learned by PTR as $\tilde{\boldsymbol{u}}_i$.

In Table 1, we list the top six topics of a sample user that are ranked based on the three vectors: $\boldsymbol{u}_i$, $\boldsymbol{\theta}_i$, and $\tilde{\boldsymbol{u}}_i$. We can see that, top topics in $\tilde{\boldsymbol{u}}_i$ are the mixtures of top topics in $\boldsymbol{u}_i$ and $\boldsymbol{\theta}_i$: topics 2, 4, and 6 come from $\boldsymbol{u}_i$, topics 3 and 5 come from $\boldsymbol{\theta}_i$, and topic 1 is shared by $\boldsymbol{u}_i$ and $\boldsymbol{\theta}_i$. This result demonstrates that PTR does a nice job of balancing the effects of users' productions and ratings, and hence it can learn better user representations and make more precise recommendations.

5 Conclusions and future work

In this paper, we have proposed a probabilistic generative model for recommending user-generated content (UGC). We have showed that the proposed model significantly improves recommendation performance by explicitly modeling the authorship information of UGC. In addition, our model provides interpretable topic representations for both users and items. Furthermore, our model helps to solve the cold-start problems.

For future work, we are interested in two directions. The first is to parallelize our algorithm and examine it on large-scale datasets. The second is to provide a Bayesian treatment of our model and develop a better learning algorithm.

Table 1 Top topics of a sample user

Vector	Top topics
Latent factors learned from CTR ($\boldsymbol{u}_i$)	1. node, propagation, graph, spread, link, random, influence, cascade, walk, diffusion, connect, edge 2. online, news, content, article, media, user, storage, popular, feed, use, collect, discuss, comment, event 3. measure, impact, subgroup, insight, factor, influence, difference, evaluation, individual, assess, forum 4. time series, time-series, subsequent, segment, distance, period, efficient, dynamic, real, transform, match 5. email, spam, filter, message, use, e-mail, twitter, detect, communication, mail, tweet, spammer, messages 6. million, large, thousand, ten, hundred, severe, scale, time, size, human, large-scale, massive, scalable
Topic proportions learned from PTR ($\boldsymbol{\theta}_i$)	1. network, social, communication, structure, networks, link, interaction, graph, relationship, communities 2. node, propagation, graph, spread, link, random, influence, cascade, walk, diffuse, large, generate 3. blog, post, weblog, blogger, author, use, blogospher, polite, topic, authorship, interest, link, communication 4. predict, use, forecast, future, model, predictor, accurate, prediction, target, behavior, potential, accuracy 5. topic, document, latent, text, Dirichlet, LDA, semantic, probabilistic, topics, allocation, corpus, generative 6. network, traffic, trace, perform, synchronize, process, packet, communication, delay, flow, protocol
Latent factors learned from PTR ($\tilde{\boldsymbol{u}}_i$)	1. node, propagation, graph, spread, link, random, influence, cascade, walk, diffuse, large, generate 2. time, series, time-series, similar, subsequent, segment, distance, match, period, efficient, dynamic 3. blog, post, weblog, blogger, author, use, blogospher, polite, topic, authorship, interest, link, communication 4. online, news, content, article, media, user, popular, system, feed, collect, discuss, comment, event, site 5. network, social, communication, structure, networks, link, interaction, graph, relationship, communities 6. measure, impact, subgroup, factor, influence, evaluation, result, individual, assess, investigation

References

Agarwal, D., Chen, B.C., 2009. Regression-based latent factor models. Proc. 15th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, p.19-28. http://dx.doi.org/10.1145/1557019.1557029

Agarwal, D., Chen, B.C., 2010. fLDA: matrix factorization through latent Dirichlet allocation. Proc. 3rd ACM Int. Conf. on Web Search and Data Mining, p.91-100. http://dx.doi.org/10.1145/1718487.1718499

Blei, D.M., Ng, A.Y., Jordan, M.I., 2003. Latent Dirichlet allocation. *J Mach. Learn. Res.*, **3**:993-1022.

Blei, D.M., Griffiths, T.L., Jordan, M.I., 2010. The nested Chinese restaurant process and Bayesian nonparametric inference of topic hierarchies. *J. ACM*, **57**(2):7. http://dx.doi.org/10.1145/1667053.1667056

de Pessemier, T., Deryckere, T., Martens, L., 2011. Context aware recommendations for user-generated content on a social network site. Proc. 7th European Interactive Television Conf., p.133-136. http://dx.doi.org/10.1145/1542084.1542108

Deshpande, M., Karypis, G., 2004. Item-based top-*N* recommendation algorithms. *ACM Trans. Inform. Syst.*, **22**(1): 143-177. http://dx.doi.org/10.1145/963770.963776

Duchi, J., Shalev-Shwartz, S., Singer, Y., *et al.*, 2008. Efficient projections onto the ℓ_1-ball for learning in high dimensions. Proc. 25th Int. Conf. on Machine Learning, p.272-279.

Hu, Y., Koren, Y., Volinsky, C., 2008. Collaborative filtering for implicit feedback datasets. 8th IEEE Int. Conf. on Data Mining, p.263-272. http://dx.doi.org/10.1109/ICDM.2008.22

Li, Y.M., Yang, M., Zhang, Z.F., 2013. Scientific articles recommendation. Proc. 22nd ACM Int. Conf. on Information and Knowledge Management, p.1147-1156. http://dx.doi.org/10.1145/2505515.2505705

Linden, G., Smith, B., York, J., 2003. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Intern. Comput.*, **7**(1):76-80. http://dx.doi.org/10.1109/MIC.2003.1167344

Lops, P., de Gemmis, M., Semeraro, G., 2011. Content-based recommender systems: state of the art and trends. *In*: Ricci, F., Rokach, L., Shapira, B., *et al.* (Eds.), Recommender Systems Handbook. Springer, Boston, p.73-105. http://dx.doi.org/10.1007/978-0-387-85820-3_3

Melville, P., Mooney, R.J., Nagarajan, R., 2002. Content-boosted collaborative filtering for improved recommendations. Proc. 8th National Conf. on Artificial Intelligence, p.187-192.

Mooney, R.J., Roy, L., 2000. Content-based book recommending using learning for text categorization. Proc. 5th ACM Conf. on Digital Libraries, p.195-204. http://dx.doi.org/10.1145/336597.336662

Pan, R., Zhou, Y., Cao, B., *et al.*, 2008. One-class collaborative filtering. 8th IEEE Int. Conf. on Data Mining, p.502-511. http://dx.doi.org/10.1109/ICDM.2008.16

Purushotham, S., Liu, Y., Kuo, C.C.J., 2012. Collaborative topic regression with social matrix factorization for recommendation systems. arXiv:1206.4684.

Rendle, S., Freudenthaler, C., Gantner, Z., *et al.*, 2009. BPR: Bayesian personalized ranking from implicit feedback. Proc. 25th Conference on Uncertainty in Artificial Intelligence, p.452-461.

Salakhutdinov, R., Mnih, A., 2007. Probabilistic matrix factorization. Neural Information Processing Systems, p.1257-1264.

Salakhutdinov, R., Mnih, A., 2008. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. Proc. 25th Int. Conf. on Machine Learning, p.880-887. http://dx.doi.org/10.1145/1390156.1390267

Teh, Y.W., Jordan, M.I., Beal, M.J., *et al.*, 2004. Sharing clusters among related groups: hierarchical Dirichlet processes. Neural Information Processing Systems, p.1385-1392.

Veeramachaneni, S., Sona, D., Avesani, P., 2005. Hierarchical Dirichlet model for document classification. Proc. 22nd Int. Conf. on Machine Learning, p.928-935. http://dx.doi.org/10.1145/1102351.1102468

Wang, C., Blei, D.M., 2011. Collaborative topic modeling for recommending scientific articles. Proc. 17th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, p.448-456. http://dx.doi.org/10.1145/2020408.2020480

Xu, Y., Yin, J., 2015. Collaborative recommendation with user generated content. *Eng. Appl. Artif. Intel.*, **45**(C):281-294. http://dx.doi.org/10.1016/j.engappai.2015.07.012

Xu, Y., Chen, Z., Yin, J., *et al.*, 2015. Learning to recommend with user generated content. Int. Conf. on Web-Age Information Management, p.221-232. http://dx.doi.org/10.1007/978-3-319-21042-1_18