Collaborative Topic Modeling for Recommending Scientific Articles

Chong Wang
Computer Science Department
Princeton University
Princeton, NJ, 08540, USA
chongw@cs.princeton.edu

David M. Blei
Computer Science Department
Princeton University
Princeton, NJ, 08540, USA
blei@cs.princeton.edu

ABSTRACT

Researchers have access to large online archives of scientific articles. As a consequence, finding relevant papers has become more difficult. Newly formed online communities of researchers sharing citations provides a new way to solve this problem. In this paper, we develop an algorithm to recommend scientific articles to users of an online community. Our approach combines the merits of traditional collaborative filtering and probabilistic topic modeling. It provides an interpretable latent structure for users and items, and can form recommendations about both existing and newly published articles. We study a large subset of data from CiteULike, a bibliography sharing service, and show that our algorithm provides a more effective recommender system than traditional collaborative filtering.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*information filtering*; I.2.6 [**Artificial Intelligence**]: Learning—*Parameter learning*

General Terms

Algorithms, Experimentation, Performance

Keywords

Scientific article recommendation, Topic modeling, Collaborative filtering, Latent structure interpretation

1. INTRODUCTION

Modern researchers have access to large archives of scientific articles. These archives are growing as new articles are placed online and old articles are scanned and indexed. While this growth has allowed researchers to quickly access more scientific information, it has also made it more difficult for them to find articles relevant to their interests. Modern researchers need new tools for managing what is available to them.

Historically, one way that researchers find articles is by following citations in other articles that they are interested in. This is an effective practice—and one that we should continue—but it limits researchers to specific citation communities, and it is biased towards heavily cited papers. A statistician may miss a relevant paper in economics or biology because the two literatures rarely cite each other; and she may miss a relevant paper in statistics because it was also missed by the authors of the papers that she has read. One of the opportunities of online archives is to inform researchers about literature that they might not be aware of.

A complementary method of finding articles is keyword search. This is a powerful approach, but it is also limited. Forming queries for finding new scientific articles can be difficult as a researcher may not know what to look for; search is mainly based on content, while good articles are also those that many others found valuable; and search is only good for directed exploration, while many researchers would also like a “feed” of new and interesting articles.

Recently, websites like CiteULike[1] and Mendeley[2] allow researchers to create their own reference libraries for the articles they are interested in and share them with other researchers. This has opened the door to using recommendation methods [13] as a third way to help researchers find interesting articles. In this paper, we develop an algorithm for recommending scientific articles to users of online archives. Each user has a library of articles that he or she is interested in, and our goal is to match each user to articles of interest that are not in his or her library.

We have several criteria for an algorithm to recommend scientific articles. First, recommending older articles is important. Users of scientific archives are interested in older articles for learning about new fields and understanding the foundations of their fields. When recommending old articles, the opinions of other users plays a role. A foundational article will be in many users’ libraries; a less important article will be in few.

Second, recommending new articles is also important. For example, when a conference publishes its proceedings, users would like see the recommendations from these new articles to keep up with the state-of-the-art in their discipline. Since the articles are new, there is little information about which or how many other users placed the articles in their libraries, and thus traditional collaborative filtering methods has difficulties making recommendations. With new articles, a recommendation system must use their content.

Finally, exploratory variables can be valuable in online scientific archives and communities. For example, we can summarize and describe each user’s preference profile based on the content of the articles that he or she likes. This lets us connect similar users to enhance the community, and indicate why we are connecting them. Further, we can describe articles in terms of what kinds of users like them. For example, we might detect that a machine learning

KDD’11, August 21–24, 2011, San Diego, California, USA.
Copyright 2011 ACM 978-1-4503-0813-7/11/08 ...$10.00.

[1] `http://www.citeulike.org`
[2] `http://www.mendeley.com`

article is of strong interest to computer vision researchers. If enough researchers use such services, these variables might also give an alternative measure of the impact of an article within a field.

With these criteria in mind, we develop a machine learning algorithm for recommending scientific articles to users in an online scientific community. Our algorithm uses two types of data—the other users' libraries and the content of the articles—to form its recommendations. For each user, our algorithm can finds both older papers that are important to other similar users and newly written papers whose content reflects the user's specific interests. Finally, our algorithm gives interpretable representations of users and articles.

Our approach combines ideas from collaborative filtering based on latent factor models [17, 18, 13, 1, 22] and content analysis based on probabilistic topic modeling [7, 8, 20, 2]. Like latent factor models, our algorithm uses information from other users' libraries. For a particular user, it can recommend articles from other users who liked similar articles. Latent factor models work well for recommending known articles, but cannot generalize to previously unseen articles.

To generalize to unseen articles, our algorithm uses topic modeling. Topic modeling provides a representation of the articles in terms of latent themes discovered from the collection. When used in our recommender system, this component can recommend articles that have similar content to other articles that a user likes. The topic representation of articles allows the algorithm to make meaningful recommendations about articles before anyone has rated them.

We combine these approaches in a probabilistic model, where making a recommendation for a particular user is akin to computing a conditional expectation of hidden variables. We will show how the algorithm for computing these expectations naturally balances the influence of the content of the articles and the libraries of the other users. An article that has not been seen by many will be recommended based more on its content; an article that has been widely seen will be recommended based more on the other users.

We studied our algorithm with data from CiteULike: $5,551$ users, $16,980$ articles, and $204,986$ bibliography entries. We will demonstrate that combining content-based and collaborative-based methods works well for recommending scientific articles. Our method provides better performance than matrix factorization methods alone, indicating that content can improve recommendation systems. Further, while traditional collaborative filtering cannot suggest articles before anyone has rated them, our method can use the content of new articles to make predictions about who will like them.

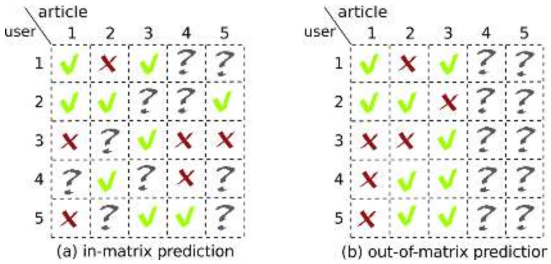
2. BACKGROUND

We first give some background. We describe two types of recommendation problems we address; we describe the classical matrix factorization solution to recommendation; and we review latent Dirichlet allocation (LDA) for topic modeling of text corpora.

2.1 Recommendation Tasks

The two elements in a recommender system are users and items. In our problem, items are scientific articles and users are researchers. We will assume I users and J items. The rating variable $r_{ij} \in \{0, 1\}$ denotes whether user i includes article j in her library [12]. If it is in the library, this means that user i is interested in article j. (This differs from some other systems where users explicitly rate items on a scale.) Note that $r_{ij} = 0$ can be interpreted into two ways. One way is that user i is not interested in article j; the other is that user i does not know about article j.

For each user, our task is to recommend articles that are not in her library but are potentially interesting. There are two types of recommendation: *in-matrix prediction* and *out-of-matrix prediction*. Figure 1 illustrates the idea.



user \ article	1	2	3	4	5
1	✓	×	✓	?	?
2	✓	✓	?	?	✓
3	×	?	✓	×	×
4	?	✓	?	×	?
5	×	?	✓	✓	?

(a) in-matrix prediction

user \ article	1	2	3	4	5
1	✓	×	✓	?	?
2	✓	✓	×	?	?
3	×	×	✓	?	?
4	×	✓	✓	?	?
5	×	✓	✓	?	?

(b) out-of-matrix predictior

Figure 1: Illustration of the two tasks for scientific article recommendation systems, where $\surd$ indicates "like", $\times$ "dislike" and ? "unknown".

In-matrix prediction. Figure 1 (a) illustrates in-matrix prediction. This refers to the problem of making recommendations about those articles that have been rated by at least one user in the system. This is the task that traditional collaborative filtering can address.

Out-of-matrix prediction. Figure 1 (b) illustrates out-of-matrix prediction, where articles 4 and 5 have never been rated. (This is sometimes called "cold start recommendation.") Traditional collaborative filtering algorithms cannot make predictions about these articles because those algorithms only use information about other users' ratings. This task is important for online scientific archives, however, because users want to see new articles in their fields. A recommender system that cannot handle out-of-matrix prediction cannot recommend newly published papers to its users.

2.2 Recommendation by Matrix Factorization

The traditional approach to recommendation is collaborative filtering (CF), where items are recommended to a user based on other users with similar patterns of selected items. (Note that collaborative filtering does not use the content of the items.) Most successful recommendation methods are *latent factor models* [17, 18, 13, 1, 22], which provide better recommendation results than the *neighborhood methods* [11, 13]. In this paper, we focus on latent factor models.

Among latent factor methods, matrix factorization performs well [13]. In matrix factorization, we represent users and items in a shared latent low-dimensional space of dimension K—user i is represented by a latent vector $u_i \in \mathbb{R}^K$ and item j by a latent vector $v_j \in \mathbb{R}^K$. We form the prediction of whether user i will like item j with the inner product between their latent representations,

$$\hat{r}_{ij} = u_i^T v_j. \tag{1}$$

Biases for different users and items can also be incorporated [13].

To use matrix factorization, we must compute the latent representations of the users and items given an observed matrix of ratings. The common approach is to minimize the regularized squared error loss with respect to $U = (u_i)_{i=1}^I$ and $V = (v_j)_{j=1}^J$,

$$\min_{U,V} \textstyle\sum_{i,j} (r_{ij} - u_i^T v_j)^2 + \lambda_u ||u_i||^2 + \lambda_v ||v_j||^2, \tag{2}$$

where λ_u and λ_v are regularization parameters.

This matrix factorization for collaborative filtering can be generalized as a probabilistic model [18]. In probabilistic matrix factorization (PMF), we assume the following generative process,

1. For each user i, draw user latent vector $u_i \sim \mathcal{N}(0, \lambda_u^{-1} I_K)$.
2. For each item j, draw item latent vector $v_j \sim \mathcal{N}(0, \lambda_v^{-1} I_K)$.

3. For each user-item pair (i, j), draw the response

$$r_{ij} \sim \mathcal{N}(u_i^T v_j, c_{ij}^{-1}), \tag{3}$$

where c_{ij} is the precision parameter for r_{ij}.

(Note that I_K is a K-dimensional identity matrix.) This is the interpretation of matrix factorization that we will build on.

When $c_{ij} = 1$, for $\forall i, j$, the maximum a posteriori estimation (MAP) of PMF corresponds to the solution in Eq. 2. Here, the precision parameter c_{ij} serves as a confidence parameter for rating r_{ij}. If c_{ij} is large, we trust r_{ij} more. As we mentioned above, $r_{ij} = 0$ can be interpreted into two ways—the user i is either not interested in item j or is unaware of it. This is thus a "one-class collaborative filtering problem," similar to the TV program and news article recommendation problems studied in [12] and [16]. In that work, the authors introduce different confidence parameters c_{ij} for different ratings r_{ij}. We will use the same strategy to set c_{ij} a higher value when $r_{ij} = 1$ than when $r_{ij} = 0$,

$$c_{ij} = \begin{cases} a, & \text{if } r_{ij} = 1, \\ b, & \text{if } r_{ij} = 0, \end{cases} \tag{4}$$

where a and b are tuning parameters satisfying $a > b > 0$.

We fit a CF model by finding a locally optimal solution of the user variables U and item variables V, usually with an iterative algorithm [12]. We then use Eq. 1 to predict the ratings of the articles outside of each user's library.

There are two main disadvantages to matrix factorization for recommendation. First, the learnt latent space is not easy to interpret; second, as mentioned, matrix factorization only uses information from other users—it cannot generalize to completely unrated items.

2.3 Probabilistic Topic Models

Topic modeling algorithms [5] are used to discover a set of "topics" from a large collection of documents, where a topic is a distribution over terms that is biased around those associated under a single theme. Topic models provide an interpretable low-dimensional representation of the documents [8]. They have been used for tasks like corpus exploration, document classification, and information retrieval. Here we will exploit the discovered topic structure for recommendation.

The simplest topic model is latent Dirichlet allocation (LDA) [7]. Assume there are K topics $\boldsymbol{\beta} = \beta_{1:K}$, each of which is a distribution over a fixed vocabulary. The generative process of LDA is as follows. For each article $\boldsymbol{w}_j$ in the corpus,

1. Draw topic proportions $\theta_j \sim \text{Dirichlet}(\alpha)$.
2. For each word n,
   (a) Draw topic assignment $z_{jn} \sim \text{Mult}(\theta_j)$.
   (b) Draw word $w_{jn} \sim \text{Mult}(\beta_{z_{jn}})$.

This process reveals how the words of each document are assumed to come from a mixture of topics: the topic proportions are document-specific, but the set of topics is shared by the corpus.

Given a collection, the posterior distribution (or maximum likelihood estimate) of the topics reveals the K topics that likely generated its documents. Unlike a clustering model, where each document is assigned to one cluster, LDA allows documents to exhibit multiple topics. For example, LDA can capture that one article might be about biology and statistics, while another might be about biology and physics. Since LDA is unsupervised, the themes of "physics" "biology" and "statistics" can be discovered from the corpus; the mixed-membership assumptions lead to sharper estimates of word co-occurrence patterns.

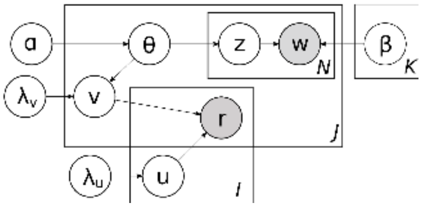


Figure 2: The graphical model for the CTR model.

Given a corpus of documents, we can use variational EM to learn the topics and decompose the documents according to them [7]. Further, given a new document, we can use variational inference to situate its content in terms of the topics. Our goal is to use topic modeling to give a content-based representation of items in a recommender system.

3. COLLABORATIVE TOPIC REGRESSION

In this section, we describe the collaborative topic regression (CTR) model. CTR combines traditional traditional collaborative filtering with topic modeling.

A first approach to combining collaborative filtering and topic modeling is to fit a model that uses the latent topic space to explain both the observed ratings and the observed words. For example, we can use the topic proportion θ_j in place of the latent item latent vector v_j in Eq. 3,

$$r_{ij} \sim \mathcal{N}(u_i^T \theta_j, c_{ij}^{-1}). \tag{5}$$

(We note that [19] proposed a similar approach to Eq. 5, but based on correlated topic models [4]. It showed modest improvement over matrix factorization on several movie recommendation datasets.)

This model suffers from the limitation that it cannot distinguish topics for explaining recommendations from topics important for explaining content. Consider two articles A and B that are both about machine learning applied to social networks. They are similar and, therefore, have similar topic proportions θ_A and θ_B. Now further suppose that these articles are interesting to different kinds of users: Article A might give an interesting machine learning algorithm that is applied to social network applications; article B uses standard machine learning techniques, but gives an important piece of data analysis on social network data.

Users that work in machine learning will prefer article A and rarely consider article B; users that work in social networks will prefer the opposite. However, using the topic proportions as in Eq. 5 will be likely to make similar recommendations for both articles to both types of users. Collaborative topic regression can detect this difference—that one type of user likes the first article and another type likes the second.

As above, collaborative topic regression (CTR) represents users with topic interests and assumes that documents are generated by a topic model. CTR additionally includes a latent variable ϵ_j that offsets the topic proportions θ_j when modeling the user ratings. As more users rate articles, we have a better idea of what this offset is. This offset variable can explain, for example, that article A is more interesting to machine learning researchers than it is to social network analysis researchers. How much of the prediction relies on content and how much it relies on other users depends on how many users have rated the article.

Figure 2 shows the graphical model. Again, assume there are K topics $\boldsymbol{\beta} = \beta_{1:K}$. The generative process of CTR is as follows,

1. For each user i, draw user latent vector $u_i \sim \mathcal{N}(0, \lambda_u^{-1} I_K)$.

2. For each item j,
   - (a) Draw topic proportions $\theta_j \sim \text{Dirichlet}(\alpha)$.
   - (b) Draw item latent offset $\epsilon_j \sim \mathcal{N}(0, \lambda_v^{-1} I_K)$ and set the item latent vector as $v_j = \epsilon_j + \theta_j$.
   - (c) For each word w_{jn},
     - i. Draw topic assignment $z_{jn} \sim \text{Mult}(\theta)$.
     - ii. Draw word $w_{jn} \sim \text{Mult}(\beta_{z_{jn}})$.
3. For each user-item pair (i, j), draw the rating

$$r_{ij} \sim \mathcal{N}(u_i^T v_j, c_{ij}^{-1}). \tag{6}$$

The key property in CTR lies in how the item latent vector v_j is generated. Note that $v_j = \epsilon_j + \theta_j$, where $\epsilon_j \sim \mathcal{N}(0, \lambda_v^{-1} I_k)$, is equivalent to $v_j \sim \mathcal{N}(\theta_j, \lambda_v^{-1} I_K)$, where we assume the item latent vector v_j is close to topic proportions θ_j, but could diverge from it if it has to. Note that the expectation of r_{ij} is a linear function of θ_j,

$$\mathbb{E}[r_{ij} | u_i, \theta_j, \epsilon_j] = u_i^T (\theta_j + \epsilon_j).$$

This is why we call the model collaborative topic regression.

Learning the parameters. Given topic parameter $\boldsymbol{\beta}$, computing the full posterior of u_i, v_j and θ_j is intractable. We develop an EM-style algorithm to learn the maximum a posteriori (MAP) estimates.

Maximization of the posterior is equivalent to maximizing the complete log likelihood of U, V, $\theta_{1:J}$, and R given λ_u, λ_v and β,

$$\begin{aligned} \mathcal{L} = &-\tfrac{\lambda_u}{2} \textstyle\sum_i u_i^T u_i - \tfrac{\lambda_v}{2} \textstyle\sum_j (v_j - \theta_j)^T (v_j - \theta_j) \\ &+ \textstyle\sum_j \sum_n \log \left(\sum_k \theta_{jk} \beta_{k, w_{jn}} \right) - \sum_{i,j} \tfrac{c_{ij}}{2} (r_{ij} - u_i^T v_j)^2. \end{aligned} \tag{7}$$

We have omitted a constant and set $\alpha = 1$. We optimize this function by coordinate ascent, iteratively optimizing the collaborative filtering variables $\{u_i, v_j\}$ and the topic proportions θ_j.

For u_i and v_j, maximization follows in a similar fashion as for basic matrix factorization [12]. Given the current estimate of θ_j, taking the gradient of $\mathcal{L}$ with respect to u_i and v_j and setting it to zero leads to (recall the matrix definition $U = (u_i)_{i=1}^I$ and $V = (v_j)_{j=1}^J$)

$$u_i \leftarrow (V C_i V^T + \lambda_u I_K)^{-1} V C_i R_i \tag{8}$$

$$v_j \leftarrow (U C_j U^T + \lambda_v I_K)^{-1} (U C_j R_j + \lambda_v \theta_j). \tag{9}$$

where C_i is a diagonal matrix with $c_{ij}, j = 1 \cdots, J$ as its diagonal elements and $R_i = (r_{ij})_{j=1}^J$ for user i. For item j, C_j and R_j are similarly defined. Eq. 9 shows how topic proportions θ_j affects item latent vector v_j, where λ_v balances this effect. Finally, we note that the complexity is linear in the number of articles in the users' libraries. This follows from the special structure of c_{ij} defined in Eq. 4. (See [12] for details.)

Given U and V, we now describe how to learn the topic proportions θ_j.[3] We first define $q(z_{jn} = k) = \phi_{jnk}$. Then we separate the items that contain θ_j and apply Jensen's inequality,

$$\begin{aligned} \mathcal{L}(\theta_j) &\geq -\tfrac{\lambda_v}{2} (v_j - \theta_j)^T (v_j - \theta_j) \\ &\quad + \textstyle\sum_n \sum_k \phi_{jnk} \left(\log \theta_{jk} \beta_{k, w_{jn}} - \log \phi_{jnk} \right) \\ &= \mathcal{L}(\theta_j, \boldsymbol{\phi}_j). \end{aligned} \tag{10}$$

Let $\boldsymbol{\phi}_j = (\phi_{jnk})_{n=1,k=1}^{N \times K}$. The optimal ϕ_{jnk} satisfies

$$\phi_{jnk} \propto \theta_{jk} \beta_{k, w_{jn}}. \tag{11}$$

The $\mathcal{L}(\theta_j, \boldsymbol{\phi}_j)$ gives the *tight* lower bound of $\mathcal{L}(\theta_j)$. We cannot optimize θ_j analytically, so we use projection gradient [3]. We use coordinate ascent to optimize the remaining parameters, U, V, $\boldsymbol{\theta}_{1:J}$ and $\boldsymbol{\phi}_{1:J}$.

After we estimate U, V and $\boldsymbol{\phi}$, we can optimize β,

$$\beta_{kw} \propto \textstyle\sum_j \sum_n \phi_{jnk} 1[w_{jn} = w]. \tag{12}$$

Note this is the same M-step update for topics as in LDA [7].

Prediction. After all the (locally) optimal parameters U^*, V^*, $\boldsymbol{\theta}_{1:J}^*$ and $\boldsymbol{\beta}^*$ are learned, the CTR model can be used for both in-matrix and out-of-matrix prediction. Let D be the observed data, in general each prediction is estimated as

$$\mathbb{E}[r_{ij} | D] \approx \mathbb{E}[u_i \mid D]^T \left(\mathbb{E}[\theta_j \mid D] + \mathbb{E}[\epsilon_j \mid D] \right). \tag{13}$$

For in-matrix prediction, we use the point estimate of u_i, θ_j and ϵ_j to approximate their expectations,

$$r_{ij}^* \approx (u_i^*)^T (\theta_j^* + \epsilon_j^*) = (u_i^*)^T v_j^*, \tag{14}$$

where recall that $v_j = \theta_j + \epsilon_j$.

In out-of-matrix prediction the article is new, and no other ratings are available. Thus, $\mathbb{E}[\epsilon_j] = 0$ and we predict with

$$r_{ij}^* \approx (u_i^*)^T \theta_j^*. \tag{15}$$

To obtain the topic proportions θ_j^* for a new article, we optimize Eq. 10. The first term is dropped because $v_j = \theta_j$.

Related work. Several other work uses content for recommendation [15, 14, 1, 2]. Among these, the closest work to ours is fLDA by [2]. FLDA generalizes the supervised topic model (sLDA) [6], using the empirical topic proportions $\bar{z}_j = (1/N) \sum_{n=1}^N z_{jn}$ as well as several other covariates to form predictions. In our settings, where we do not have additional covariates, their approach is roughly akin to setting $v_j = \theta_j$. We show in Section 4 that a similar setting does not perform as well as the CTR model because it largely ignores the other users ratings.

Other recent work considers the related problem of using topic modeling to predict legislative votes [21, 10]. Neither of these methods introduces offset terms to account for votes (i.e., ratings). Legislative votes might be an interesting application for the CTR model.

4. EMPIRICAL STUDY

We demonstrate our model by analyzing a real-world community of researchers and their citation files.[4]

Dataset. Our data are users and their libraries of articles obtained from CiteULike.[5] At CiteUlike, registered users create personal reference libraries; each article usually has a title and abstract. (The other information about the articles, such as the authors, publications and keywords, is not used in this paper.)

We merged duplicated articles, removed empty articles, and removed users with fewer than 10 articles to obtain a data set of $5,551$ users and $16,980$ articles with $204,986$ observed user-item pairs. (This matrix has a sparsity of 99.8%; it is highly sparse.) On average, each user has 37 articles in the library, ranging from 10 to 403. 93% of the users have fewer than 100 articles.

For each article, we concatenate its title and abstract. We remove stop words and use tf-idf to choose the top $8,000$ distinct words as the vocabulary [5]. This yielded a corpus of 1.6M words. These articles were added to CiteULike between 2004 and 2010. On

[3] On our data, we found that simply fixing θ_j as the estimate from vanilla LDA gives comparable performance and saves computation.

[4] A demo of the results can be found at
http://www.cs.princeton.edu/˜chongw/citeulike/

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

average, each article appears in 12 users' libraries, ranging from 1 to 321. 97% of the articles appear in fewer than 40 libraries.

Evaluation. In our experiments, we will analyze a set of articles and user libraries. We will evaluate recommendation algorithms on sets of held-out articles and ratings. We will (hypothetically) present each user with M articles sorted by their predicted rating and evaluate based on which of these articles were actually in each user's library.

Two possible metrics are precision and recall. However, as we discussed earlier, zero ratings are uncertain. They may indicate that a user does not like an article or does not know about it. This makes it difficult to accurately compute precision. Rather, since ratings of $r_{ij} = 1$ are known to be true positives, we focus on recall. Recall only considers the positively rated articles within the top M—a high recall with lower M will be a better system. For each user, the definition of *recall*@M is

$$\text{recall@}M = \frac{\text{number of articles the user likes in top } M}{\text{total number of article the user likes}}.$$

The recall for the entire system can be summarized using the average recall from all users.

The recall above we defined is user-oriented. We also consider article-oriented recall for testing the system's predictive performance on a particular article. For article j, we consider the population of users that like the article and the proportion of those for whom that article appears in their top M recommended articles. This evaluates the predictive power of the system on a chosen set of articles.

As we discussed in section 2, we consider two recommendation tasks users, *in-matrix prediction* and *out-of-matrix prediction*.

In-matrix prediction. In-matrix prediction considers the case where each user has a set of articles that she has not seen, but that at least one other user has seen. We ask the question, how good is each system at rating that set of articles for each user?

As discussed in section 2.1, this task is similar to traditional collaborative filtering. We split the data into a training set and test set, ensuring that all articles in the test set have appeared in the training set. Content information is not required to perform recommendations—though we will see that it helps—and thus matrix factorization can be used.

We use 5-fold cross-validation. For every article that appears at least 5 times in the users' libraries, we evenly split their user-item pairs (both 1's and 0's) into 5 folds. We iteratively consider each fold to be a test set and the others to be the training set. For those articles that appear fewer than 5 times, we always put them into the training set. This guarantees that all articles in the test set must appear in the training set. (9% of the articles are always in the training set, since they appear fewer than 5 times.)

For each fold, we fit a model to the training set and test on the within-fold articles for each user. (Note: each user has a different set of within-fold articles.) We form predictive ratings for the test set, and generate a list of the top M recommended articles.

Out-of-matrix prediction. Out-of-matrix prediction considers the case where a new set of articles is published and no one has seen them. Again we ask, how good is each system at rating that set of articles for each user?

We again use 5-fold cross validation. First, we evenly group all articles into 5 folds. For each fold, we fit the model to the submatrix formed by the out-of-fold articles and then test the recommendations for each user on the within-fold articles. Note that in this case, each user has the same set of within-fold articles and we are guaranteed that none of these articles is in the training set for any user. Again, we form predictive ratings for the test set, and generate a list of the top M recommended articles.

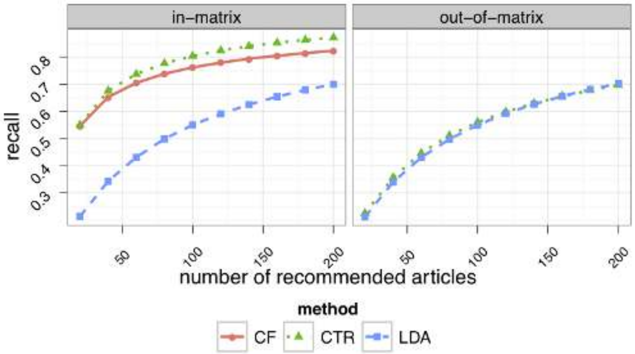


Figure 3: Recall comparison on in-matrix and out-of-matrix prediction tasks by varying the number of recommended articles. For CTR, we set $\lambda_v = 100$. Error bars are too small to show. The maximum expected recall for random recommendation is about 6%. CF can not do out-of-matrix prediction. CTR performs best.

These two experimental set-ups—in-matrix and out-of-matrix predictions—are designed to be comparable—the top M articles are computed from the same size of candidate populations.

Experimental settings. For matrix factorization for collaborative filtering (CF), we used grid search to find that $K = 200$, $\lambda_u = \lambda_v = 0.01$, $a = 1$, $b = 0.01$ gives good performance on held out recommendations. We use CF to denote this method.

For collaborative topic regression (CTR), we set the parameters similarly as for CF, $K = 200$, $\lambda_u = 0.01$, $a = 1$ and $b = 0.01$. In addition, the precision parameter λ_v balances how the article's latent vector v_j diverges from the topic proportions θ_j. We vary $\lambda_v \in \{10, 100, 1000, 10000\}$, where a larger λ_v increases the penalty of v_j diverging from θ_j.

We also compare to the model that only uses LDA-like features, as we discussed in the beginning of section 3. This is equivalent to fixing the per-item latent vector $v_j = \theta_j$ in the CTR model. This is a nearly content-only model—while the per-user vectors are fit to the ratings data, the document vectors θ_j are only based on the words of the document. We use LDA to denote this method. (Note that we use the resulting topics and proportions of LDA to initialize the CTR model.)

The baseline is the random model, where a user see M random recommended articles. We note that the expected recall for the random method from a pool of M_{tot} articles is irrelevant to library size. It is always M/M_{tot}.

Comparisons. Figure 3 shows the overall performance for in-matrix and out-of-matrix prediction, when we vary the number of returned articles $M = 20, 40, \cdots, 200$. For CTR, we pick $\lambda_v = 100$; Figure 4 shows the performs when we change λ_v for the CTR model compared with CF and LDA when we fix $M = 100$.

Figure 3 and 4 shows that matrix factorization works well for in-matrix prediction, but adding content with CTR improves performance. The improvement is greater when the number of returned documents M is larger. The reason is as follows. Popular articles are more likely to be recommended by both methods. However, when M becomes large, few user ratings are available to ensure that CF gives good recommendations; the contribution of the content becomes more important.

Compared to both CF and CTR, LDA suffers for in-matrix pre-

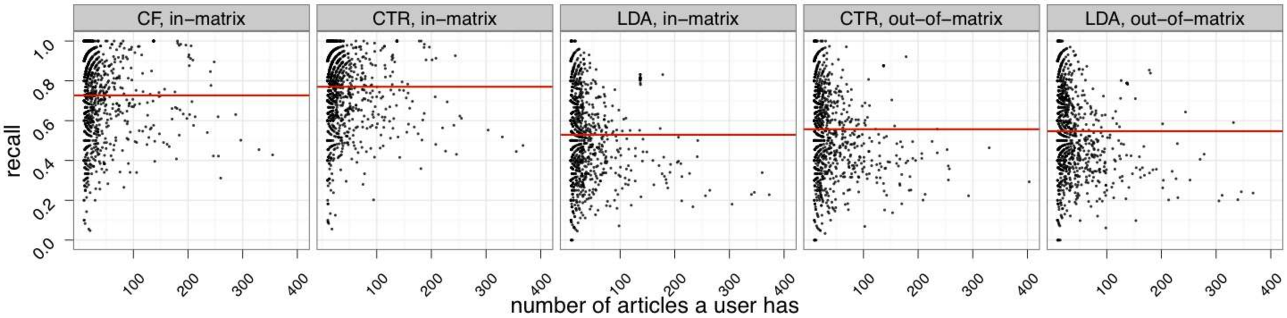


Figure 5: These scatter plots show how the number of articles a user has affects his or her recall. Red lines indicate the average. In these plots, the number of recommended articles is 100. CF can not do out-of-matrix prediction. This shows that CTR performs the best over user-oriented recall.

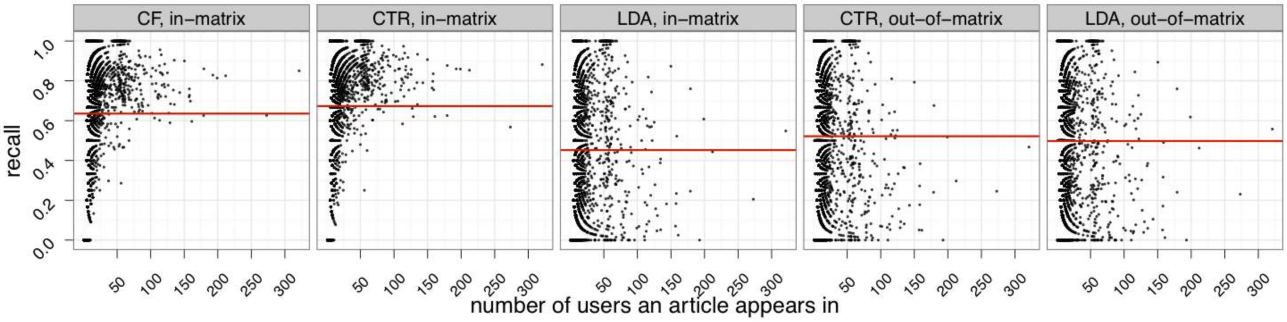


Figure 6: These scatter plots show how the number of users that like an article affects its recall. Red lines indicate the average. In these plots, the number of recommended articles is 100. CF can not do out-of-matrix prediction. This shows that CTR performs best over article-oriented recall as well.

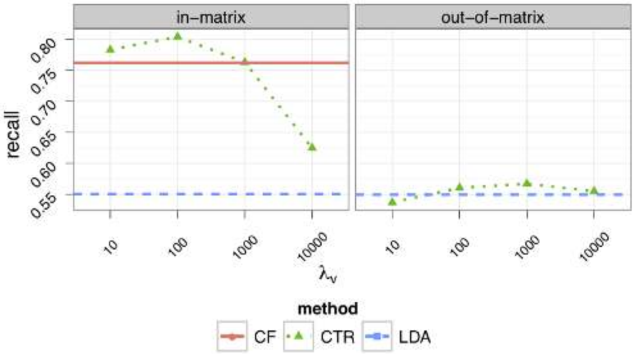


Figure 4: Recall comparison on in-matrix and out-of-matrix prediction tasks by fixing the number of recommended articles at $M = 100$. Error bars are too small to show. This shows how the precision parameter λ_v affects the performance of CTR. The expected recall of random recommendation is about 3%. CF can not do out-of-matrix prediction.

diction. It does not account enough for the users' information in forming its predicted ratings. The gap between CF and LDA is interesting—other users provide a better assessment of preferences than content alone.

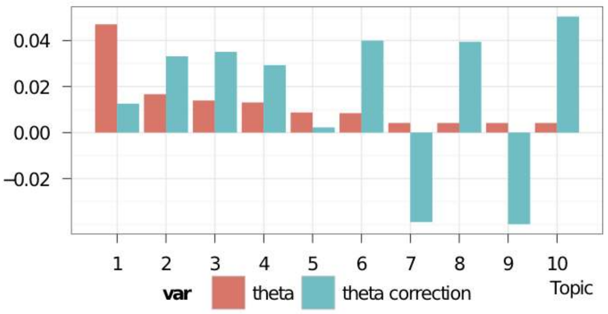
Out-of-matrix prediction is a harder problem, as shown by the relatively lower recall. In this task, CTR performs slightly better than LDA. Matrix factorization cannot perform out-of-matrix prediction. (Note also that LDA performs almost the same on both in-matrix and out-of-matrix predictions. This is expected because, in both settings, it makes its recommendations almost entirely based on content.) Overall, CTR is the best model.

In Figure 4 we study the effect of the precision parameter λ_v. When λ_v is small in CTR, the per-item latent vector v_j can diverge significantly from the topic proportions θ_j. Here, CTR behaves more like matrix factorization where no content is considered. When λ_v increases, CTR is penalized for v_j diverging from the topic proportions; this brings the content into the recommendations. When λ_v is too large, v_j is nearly the same as θ_j and, consequently, CTR behaves more like LDA.

We next study the relationship, across models, between recommendation performance and properties of the users and articles. For this study we set the number of recommended articles $M = 100$ and the precision $\lambda_v = 100$. Figure 5 shows how the performance varies as a function of the number of articles in a user's library; Figure 6 shows how the performance varies as a function of the number of users that like an article.

As we see from Figure 5, for both in-matrix and out-of-matrix prediction, users with more articles tend to have less variance in their predictions. Users with few articles tend to have a diversity in the predictions, whose recall values vary around the extreme values of 0 and 1. In addition, we see that recall for users with more articles have a decreasing trend. This is reasonable because when a user has more articles then there will be more infrequent ones. As we see next, these articles are harder to predict.

From Figure 6, on in-matrix prediction for CF, CTR and LDA articles with high frequencies tend to have high recalls for in-matrix prediction and their predictions have less variance. This is because these articles have more collaborative information than infrequent ones, and, furthermore, CF and CTR make use of this information.



topic 1: estimate, estimates, likelihood, maximum, estimated, missing, distances
topic 10: parameters, Bayesian, inference, optimal, procedure, prior, assumptions

Figure 7: *Maximum likelihood from incomplete data via the EM algorithm***. Here, "theta" denotes θ_j and "theta correction" denotes the offset ϵ_j. The 10 topics are obtained by joining the top 5 topics ranked by θ_{jk} and another top 5 topics ranked by $|\epsilon_{jk}|$, $k = 1, \cdots, K$. Under CTR, an article of wide interest is likely to exhibit more topics than its text exhibits. For example, this article brings in several other topics, including one on "Bayesian statistics" (topic 10). Note that the EM article is mainly about parameter estimation (topic 1), though is frequently referenced by Bayesian statisticians (and scholars in other fields as well).**

For LDA, this trend is much smaller. In out-of-matrix predictions, since predictions are made on new articles, these frequencies do not have an effect on training the model.

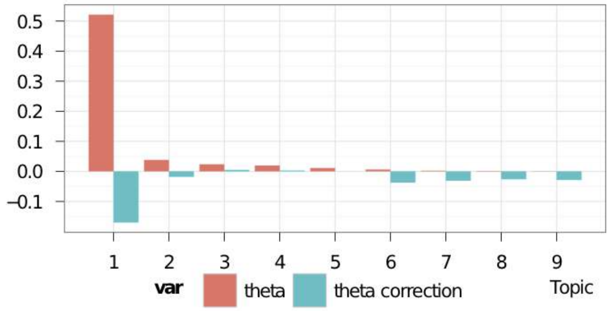
We now turn to an exploratory analysis of our results on the CTR model. (In the following, the precision $\lambda_v = 100$.)

Examining User Profiles. One advantage of the CTR model is that it can explain the user latent space using the topics learned from the data. For one user, we can find the top matched topics by ranking the entries of her latent vector u_i. Table 1 shows two example users and their top 3 matched topics along with their top 10 preferred articles as predicted by the CTR model.

The learned topics serve as a summary of what users might be interested in. For user I, we see that he or she might be a researcher working on machine learning and its applications to texts and images. Although the predicted top 10 articles don't contain a vision article, we see such articles when more articles are retrieved. For user II, he or she might be a researcher who is interested in user interfaces and collaborative filtering.

Examining the Latent Space of Articles. We can also examine the latent space of articles beyond their topic proportions. Here we inspect the articles with the largest overall offsets ϵ_j. Table 2 shows the top 10 articles with the largest offsets measured by the distance between v_j and θ_j, $\epsilon_j^T \epsilon_j = (v_j - \theta_j)^T (v_j - \theta_j)$. The last two columns show the average of predicted ratings over those users who actually have that article (avg–like) and those users who do not have that article (avg–dislike).

These articles are popular in this data. Among the top 50 articles by this measure, 94% of them have at least 50 appearances. Articles with large offsets enjoy readership from different areas, and their item latent vectors have to diverge from the topic proportions to account for this. For example, Figure 7 illustrates the article that is the main citation for the expectation-maximization algorithm, "*Maximum likelihood from incomplete data via the EM algorithm*" [9]. Its top topic (found by $k = \arg\max_k \theta_{jk}$), is shown as topic 1. It is about "parameter estimation," which is the main focus of this article. We can also examine the topics that are offset the most, $k = \arg\max_k |\epsilon_{jk}| = \arg\max_k |v_{jk} - \theta_{jk}|$. The maximum offset is for topic 10, a topic about "Bayesian statistics." Topic 10 has a low value in θ_j—the EM paper is not a Bayesian paper—but readers of Bayesian statistics typically have this paper in their library.



topic 1: neurons, responses, neuronal, spike, cortical, stimuli, stimulus

Figure 8: *Phase-of-firing coding of natural visual stimuli in primary visual cortex.* **This figure was created in the same way as Figure 7. It shows that a less popular article might also have a high offset value ϵ_j. In this case, it changes the actual magnitudes in θ_j, but does not bring in other topics.**

Examining the offset can yield the opposite kind of article. For example, consider the article "*Phase-of-firing coding of natural visual stimuli in primary visual cortex*" in Figure 8. Its most probable topic is topic 1 (about "Computational Neuroscience"). Taking into account the offset, the most probable topic does not change and nor are new topics brought in. This indicates that the offset ϵ_j only adjusts v_j so that the objective function is well minimized. This article is not as interesting to users outside of Neuroscience.

5. CONCLUSIONS AND FUTURE WORK

We proposed an algorithm for recommending scientific articles to users based on both content and other users' ratings. Our study showed that this approach works well relative to traditional matrix factorization methods and makes good predictions on completely unrated articles.

Further, our algorithm provides interpretable user profiles. Such profiles could be useful in real-world recommender systems. For example, if a particular user recognizes her profile as representing different topics, she can choose to "hide" some topics when seeking recommendations about a subject.

Acknowledgements. The authors thank anonymous reviewers for their insightful comments. Chong Wang is supported by Google PhD fellowship. David M. Blei is supported by ONR 175-6343, NSF CAREER 0745520, AFOSR 09NL202, the Alfred P. Sloan foundation, and a grant from Google.

6. REFERENCES

[1] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 19–28, New York, NY, USA, 2009. ACM.

[2] D. Agarwal and B.-C. Chen. flda: matrix factorization through latent Dirichlet allocation. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 91–100, New York, NY, USA, 2010. ACM.

	user I	in user's lib?
top 3 topics	1. image, measure, measures, images, motion, matching, transformation, entropy, overlap, computed, match	
	2. learning, machine, training, vector, learn, machines, kernel, learned, classifiers, classifier, generalization	
	3. sets, objects, defined, categories, representations, universal, category, attributes, consisting, categorization	
top 10 articles	1. Information theory inference learning algorithms	✓
	2. Machine learning in automated text categorization	✓
	3. Artificial intelligence a modern approach	×
	4. Data xmining: practical machine learning tools and techniques	×
	5. Statistical learning theory	×
	6. Modern information retrieval	✓
	7. Pattern recognition and machine learning, information science and statistics	✓
	8. Recognition by components: a theory of human image understanding	×
	9. Data clustering a review	✓
	10. Indexing by latent semantic analysis	✓
	user II	in user's lib?
top 3 topics	1. users, user, interface, interfaces, needs, explicit, implicit, usability, preferences, interests, personalized	
	2. based, world, real, characteristics, actual, exploring, exploration, quite, navigation, possibilities, dealing	
	3. evaluation, collaborative, products, filtering, product, reviews, items, recommendations, recommender	
top 10 articles	1. Combining collaborative filtering with personal agents for better recommendations	×
	2. An adaptive system for the personalized access to news	✓
	3. Implicit interest indicators	×
	4. Footprints history-rich tools for information foraging	✓
	5. Using social tagging to improve social navigation	✓
	6. User models for adaptive hypermedia and adaptive educational systems	✓
	7. Collaborative filtering recommender systems	✓
	8. Knowledge tree: a distributed architecture for adaptive e-learning	✓
	9. Evaluating collaborative filtering recommender systems	✓
	10. Personalizing search via automated analysis of interests and activities	✓

Table 1: Two example users. We show their position in latent space via their highest weighted topics. We list the top 10 preferred articles as predicted by CTR. The last column shows whether each article is in the user's library.

[3] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[4] D. Blei and J. Lafferty. A correlated topic model of Science. *Annals of Applied Statistics*, 1(1):17–35, 2007.

[5] D. Blei and J. Lafferty. Topic models. In A. Srivastava and M. Sahami, editors, *Text Mining: Theory and Applications*. Taylor and Francis, 2009.

[6] D. Blei and J. McAuliffe. Supervised topic models. In *Neural Information Processing Systems*, 2007.

[7] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, January 2003.

[8] J. Chang, J. Boyd-Graber, S. Gerrish, C. Wang, and D. Blei. Reading tea leaves: How humans interpret topic models. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 288–296, 2009.

[9] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

[10] S. M. Gerrish and D. M. Blei. Predicting legislative roll calls from text. In *Proceedings of the 28th Annual International Conference on Machine Learning*, ICML '11, 2011.

[11] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '99, pages 230–237, New York, NY, USA, 1999. ACM.

[12] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 263–272, Washington, DC, USA, 2008. IEEE Computer Society.

[13] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

[14] P. Melville, M. R., and R. Nagaraja. Content-boosted collaborative filtering for improved recommendations. In *American Association for Artificial Intelligence*, pages 187–192, 2002.

[15] R. J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 195–204, New York, NY, USA, 2000. ACM.

[16] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 502–511, Washington, DC, USA, 2008. IEEE Computer Society.

[17] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the 25th International Conference on Machine learning*, pages 880–887. ACM, 2008.

[18] R. Salakhutdinov and A. Mnih. Probabilistic matrix

title	# dataset	# Google	avg–like	avg–dislike
1. The structure and function of complex networks	212	5,192	0.909	0.052
2. Emergence of scaling in random networks	193	8,521	0.899	0.058
3. R: a language and environment for statistical computing	113	837	0.827	0.047
4. A mathematical theory of communication	129	39,401	0.817	0.062
5. Maximum likelihood from incomplete data via the EM algorithm	157	22,874	0.864	0.055
6. A tutorial on hidden Markov models and selected applications in speech recognition	135	11,929	0.822	0.048
7. The structure of collaborative tagging systems	321	648	0.903	0.055
8. Why most published research findings are false	161	713	0.846	0.049
9. Phase-of-firing coding of natural visual stimuli in primary visual cortex	8	64	1.057	-0.004
10. Defrosting the digital library bibliographic tools for the next generation web	179	37	0.840	0.042

Table 2: The top 10 articles with the largest discrepancy between their item latent vector v_j and topic proportions θ_j, measured by $(v_j - \theta_j)^T(v_j - \theta_j)$. Column 2 and 3 show how often they appear in the data, as well as the number of citations (retrieved from Google Scholar on Feb 17, 2011). Most of these articles are popular. The last two columns give the average values of "predicted" ratings over those users who have the article (avg–like) in the library and those who do not (avg–dislike).

factorization. *Advances in Neural Information Processing Systems*, 20:1257–1264, 2008.

[19] H. Shan and A. Banerjee. Generalized probabilistic matrix factorizations for collaborative filtering. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, pages 1025–1030, Washington, DC, USA, 2010. IEEE Computer Society.

[20] Y. Teh, M. Jordan, M. Beal, and D. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2007.

[21] E. Wang, D. Liu, J. Silva, D. Dunson, and L. Carin. Joint analysis of time-evolving binary matrices and associated documents. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2370–2378. 2010.

[22] K. Yu, J. Lafferty, S. Zhu, and Y. Gong. Large-scale collaborative prediction using a nonparametric random effects model. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1185–1192, New York, NY, USA, 2009. ACM.