Multimedia Systems (2016) 22:51–62
DOI 10.1007/s00530-014-0392-y

SPECIAL ISSUE PAPER

Enriching one-class collaborative filtering with content information from social media

Ting Yuan · Jian Cheng · Xi Zhang · Qinshan Liu · Hanqing Lu

Published online: 25 June 2014
© Springer-Verlag Berlin Heidelberg 2014

Abstract In recent years, recommender systems have become popular to handle the information overload problem of social media websites. The most widely used Collaborative Filtering methods make recommendations by mining users' rating history. However, users' behaviors in social media are usually implicit, where no ratings are available. This is a One-Class Collaborative Filtering (OCCF) problem with only positive examples. How to distinguish the negative examples from missing data is important for OCCF. Existing OCCF methods tackle this by the statistical properties of users' historical behavior; however, they ignored the rich content information in social media websites, which provide additional evidence for profiling users and items. In this paper, we propose to improve OCCF accuracy by exploiting the social media content information to find the potential negative examples from the missing user-item pairs. Specifically, we get a content topic feature for each user and item by probabilistic topic modeling and embed them into the Matrix Factorization model. Extensive experiments show that our algorithm can achieve better performance than the state-of-art methods.

Keywords One-Class Collaborative Filtering · Recommender system · Topic modeling · Social media

1 Introduction

The social media websites are emerging and booming recent years, such as YouTube,[1] Twitter[2] and Facebook.[3] The prevalent use of social media generates massive data at an unprecedented rate, which makes people difficult to explore the big data and even confused about what she/he really wants. To deal with the information overload of social media, recommender systems have emerged by suggesting users the potential enjoyed items. For example, the video recommendations in YouTube and the news recommendations in Yahoo!News.[4] These recommender systems offer users efficient platforms to find their favorite items in social media websites.

The most widely used recommendation methods are Collaborative Filtering (CF) approaches. Based on the assumption that similar users have similar behaviors on similar items [2, 45], CF methods aim at predicting users' interests by mining users' rating history [17, 23, 38–40]. These ratings are usually explicitly expressed in different scores, such as a 1–5 scale in Netflix,[5] where high scores

T. Yuan · J. Cheng (✉) · X. Zhang · H. Lu
National Laboratory of Pattern Recognition, Institute of Automation Chinese Academy of Sciences, Beijing 100190, China
e-mail: jcheng@nlpr.ia.ac.cn

T. Yuan
e-mail: tyuan@nlpr.ia.ac.cn

X. Zhang
e-mail: xi.zhang@nlpr.ia.ac.cn

H. Lu
e-mail: luhq@nlpr.ia.ac.cn

Q. Liu
CICE, Nanjing University of Information Science and Technology, Nanjing 210044, China
e-mail: qsliu@nuist.edu.cn

[1] http://www.youtube.com/.
[2] https://twitter.com/.
[3] http://www.facebook.com/.
[4] http://news.yahoo.com/.
[5] http://www.netflix.com/.

expressing viewing preferences and low scores expressing dislike. Through these positive (like) and negative (dislike) examples, CF methods can capture the users' tastes. However, in most social media websites, we can only observe users' implicit feedback, such as who watched what, who read what and who shared what. In these cases, we have only positive examples of what a user likes but lacks substantial evidence on which items user dislike, that is, if a user has not watched a video yet, we cannot determine whether she/he does not like the video or even not know it. Thus, most recommendations in social media can be considered as One-Class Collaborative Filtering (OCCF) problem, which has following typical characteristics: only positive examples can be observed; classes are highly imbalanced; and the vast majority of data are missing. How to model the negative examples is crucial to OCCF.

There are two intuitive strategies for traditional CF methods to handle this one-class problem: treating all missing user-item pairs as unknown [39], or treating all missing user-item pairs as negative [6]. However, both of them confused the negative examples and unlabeled positive examples together. Recently, some researches on OCCF problems focused on modeling the negative examples [19, 34, 35, 44]. They give a weight to each missing user-item pair as the probability to treat it as negative. However, most of them distinguish the negative examples by simply observing the statistical properties of historical feedback. For example, in [19, 34], they think if a user has viewed more items, those items that she/he has not viewed are more likely to be negative; if an item is viewed by less users, the missing data for this item are more likely to be negative. These assumptions are not always suitable for all the datasets, and users' implicit feedback offers little information to identify negative examples from missing data. For example, if an item is new to the system (cold-start) and no user has rated it, all the users' opinion on it will be taken as negative with high probability, which is unfair. Thus, it is limited for existing OCCF methods to model negative examples by simply considering the behavior history.

In social media websites, we naturally have much content information that can be leveraged [4, 5]. For example, the news' content, the movie's tag, the video's description and the image's visual information. The content information has been proved to provide important evidence for profiling users and items in [1, 3, 36, 48], which to some extent can represent user's taste on item. For example, if user's content profile is similar to a video's content profile, the user will be more likely to watch the video. In OCCF problems where it is hard to identify representative negative examples from implicit feedback, the rich content information of social media may be useful to assist modeling the missing negative examples. However, little work has been done on this.

In this paper, we propose to deal with the one-class recommendation problems of social media by exploiting the rich content information of social media. To distinguish the potential negative examples from missing data, we get a content topic feature for each user and item by probabilistic topic modeling [7, 18, 46]. Then, we extend the Matrix Factorization [23, 39] model of CF by incorporating the content-dissimilarity-based weighting scheme. Experiments on real-world data from CiteUlike[6] and YouTube show that the proposed method can effectively improve the recommendation performance.

The rest of the paper is organized as follows. We first discuss the related work in Sect. 2. Then, we will give some preliminaries in Sect. 3 and describe our method in Sect. 4. We analyze experimental results on two real-word social media datasets in Sect. 5. Finally, we conclude this work in Sect. 6.

2 Related work

Recommendation methods are usually classified into three categories [2]: Content-based recommendations [3, 13, 32, 36, 37], CF [17, 23, 38–40] and Hybrid approaches [11, 31, 41, 48]. Content-based methods make recommendations based on the similarity between the user and the item's content profile. In [3, 25, 32], they utilize the traditional heuristics, such as cosine similarity to measure the similarity, and recommend items whose content profile is similar to those the user liked in the past. In [36], based on the content profiles of items that were rated as "relevant" or "irrelevant" by the user, the author learns a Bayesian classifier to classify unrated items. Recently, Content-based methods have also been investigated in many recommender systems for social media. For example, [26, 28] recommend tags for the target image by considering the tags associated with its nearest neighbors based on visual content similarity. Mei et al. [30] proposes an online video recommender system using multimodal content relevance between videos and users' click-through data. However, the Content-based methods have following limitations: First, they must have enough information to build a reliable classifier and are limited by the features explicitly associated with the objects they recommend; second, they tend to recommend items that have similar content to those the user already rated, which leads to a poor diversity of recommendation.

Unlike Content-based methods, CF approaches predict users' interest by mining users' rating history. They do not

[6] http://www.citeulike.org/.

require content information and can discover interesting associations that the Content-based methods cannot. In general, CF methods are based on the fundamental assumption that similar users have similar behaviors on similar items [2, 45]. Notice that the "similar" here is different from the content similarity in Content-based methods, it refers to the similar rating preference. CF methods are mainly divided into two categories: memory-based and model-based. Memory-based methods [8, 17, 27, 40] usually search for similar users or items to produce a prediction. The similarity is computed based on rating history. They can be further categorized as user-based methods [8, 17, 20] or item-based methods [14, 27, 40], depending on whether the recommendation for a user is aggregated from users with similar preference to her/him or from items similar to those she/he already liked. However, memory-based methods are limited in handling highly sparse data since the rating similarity cannot be estimated accurately in this case.

Different from memory-based methods, the model-based methods first employ machine learning and statistical techniques to learn a prediction model from the known ratings of users, and then apply the model to do recommendation. Examples include the latent semantic models [18, 43], graphical models [21], Bayesian models [16, 52] and clustering models [22, 33]. Among different model-based methods, low-rank Matrix Factorization (MF) techniques have attracted much research attention [23, 38, 39], due to the advantages of scalability and accuracy. Based on the premise that users' tastes can be represented by a small number of factors, MF techniques learn the low-rank latent factors of users and items from the observed ratings in the user-item rating matrix and then utilize them to predict unobserved ratings. Traditional CF techniques have achieved successful results in rating prediction problems, such as Netflix's movie recommendation. However, it suffers from the well-known cold-start problem [41], where few ratings can be obtained when a new item or user enters to the system.

Hybrid approaches try to combine Content-based methods and CF approaches to remedy their limitations. Burke et al. [10] employ mixture models which build the recommendation based on a linear combination of the Content-based prediction and collaborative prediction. Schein et al. [41] propose to unify CF and Content-based evidence by probabilistic mixture of aspects. Recently, more and more works focus on social media recommendation [42, 50, 51]. Many of them are the Hybrid approaches, which mine the content information of social media as well as the historical behaviors of users to give a more accuracy recommendation. Wang et al. [49] design a joint social content recommendation framework for video recommendation in online social network. Particularly, they propose a user-content matrix update approach to fill in the cold user-video entries by making use of both the social network information and content information. Tiemann et al. [47] investigate ensemble learning methods to combine outputs of item-based Collaborative filtering and Content-based methods for music recommendation. Most of the methods mentioned above do not pay attention to the one-class recommendation problem dealing with implicit user feedback, which is common in social media.

The most typical characteristics of One-Class Collaborative Filtering (OCCF) problems are that only positive examples can be observed and classes are highly imbalanced [19]. It is crucial for OCCF to model negative examples from missing data. In prior works, there are several intuitive strategies to handle this problem. One common solution is to treat all the missing data as negative (AMAN) [6], which may bias the recommendation results because many missing data may be positive. Another solution is to treat all the missing data as unknown (AMAU) [39], which ignores the missing ones and only uses the positive ones into the CF models. Recently, some researches on OCCF problems focused on modeling the negative examples [19, 34, 35, 44]. The basic idea of them is to treat all the missing user-item pairs as negative, but give a weight to each of them as the probability to treat it as negative. However, most of them distinguish the negative examples by simply observing the statistical properties of historical feedback. For example, in [19, 34], they compute the number of items each user have rated and the number of users each item have been rated by, then determine the weight by them. specifically, they think if a user has viewed more items, those items that she/he has not viewed is more likely to be negative; if an item is viewed by less users, the missing data for this item are more likely to be negative, which is too coarse to model the negatives and is not suitable for all the datasets.

In existing OCCF methods, little has been studied to exploit the rich content information of social media to overcome the one-class recommendation problem. In this paper, we emphasize on improving OCCF by incorporating rich content information to distinguish the potential negative examples from missing user-item pairs. Specifically, our approach is a Hybrid method, where we get a content topic feature for each user and item by probabilistic topic modeling [7, 18, 46] and embed them into the Matrix Factorization model.

3 Preliminaries

In this section, we first give the problem definition formally and then introduce the Matrix Factorization method of Collaborative Filtering, which is the basis of our model.

3.1 Problem definition

The task of techniques discussed in this paper is to make recommendations dealing with the implicit feedback data in social media websites, which consist of only positive examples. Suppose we have a set of users $\mathcal{U} = \{u_1, \ldots, u_m\}$ and a set of items $\mathcal{V} = \{v_1, \ldots, v_n\}$. The users' implicit feedbacks on items are expressed in a matrix $\mathbf{R} = [R_{ij}]_{m \times n}$, where $R_{ij} = 1$ represents a positive example, $R_{ij} = 0$ represents a negative example. For example, if user u_i posts video v_j in her/his preference list, $R_{ij} = 1$, otherwise R_{ij} is either 0 or 1. Given the original implicit feedback matrix $\mathbf{R}$, which lack of negative examples, our goal was to model the missing negative examples and decompose the newly modeled matrix $\mathbf{R}$, and then recommend a top-N sorted list of items for each user, which is referred to as a top-N recommendation task in [12].

3.2 Matrix Factorization model

Matrix Factorization (MF) models [23, 24, 38, 39] have achieved excellent results in recommendation dealing with explicit ratings, such as Netflix movie recommendation. In Matrix Factorization models, they learn latent factors of the users and the items to represent their characteristics. Supposing users' tastes can be represented by these latent characteristics, ratings are predicted by the inner product of users' and items' latent factors. Let $\mathbf{U} \in \mathbf{R}^{f \times m}$ and $\mathbf{V} \in \mathbf{R}^{f \times n}$ be the user and item latent factor matrices, respectively, with column vectors U_i and V_j representing d-dimensional user-specific and item-specific latent factors of user u_i and item v_j. The user u_i's rating on item v_j is approximated as:

$$\hat{R}_{ij} = U_i^T V_j \tag{1}$$

To learn the latent factors $\mathbf{U}$ and $\mathbf{V}$, many of the MF methods, applied to explicit rating datasets, suggest minimizing the regularized squared error between the predicted ratings and the observed ratings, while avoiding overfitting through an adequate regularized model. The optimization function is as follows:

$$\min_{\mathbf{U},\mathbf{V}} \sum_{R_{i,j} \text{is observed}} (R_{ij} - U_i^T V_j)^2 + \lambda(\|U_i\|_F^2 + \|V_j\|_F^2) \tag{2}$$

where $\|\cdot\|_F$ is the Frobenius Norm of a matrix and λ is used for regularizing the model. The latent factors are often learnt by stochastic gradient descent (SGD) and Alternating Least Squares (ALS), see example in [24, 53].

In the prior works, MF models often focus on mining the historical ratings explicitly expressed in different scores, such as a 1–5 scale in Netflix, where high scores expressing preference and low scores expressing dislike. In these cases, they have positive and negative examples to efficiently learn the latent factors of users and items. However, in One-Class Collaborative Filtering setting, due to lack of negative examples, it is limited to directly apply the above algorithm to OCCF problem. In this paper, we will extend MF models by incorporating content information of social media to model the missing negative examples.

4 Our framework

In this section, we first introduce the overview of our framework, which focus on the one-class problem of social media recommendation. Then, we will talk about each step of our method in detail.

4.1 Our framework overview

The OCCF problems have following typical characteristics: Only positive examples can be observed, and the classes are highly imbalanced. To address the problems of OCCF, we propose to exploit the content information of social media to assist modeling negative examples, and extend the Matrix Factorization (MF) model by incorporating the content-dissimilarity based weighting scheme. Our framework consists of following steps, which is shown in Fig. 1:

Step 1: *We extract content topic feature for each user and item*. Since we attempt to utilize the content information to help us distinguish the potential negative examples from missing user-item pairs, we first analyze the content information of social media. In this step, we aim to find the useful types of content information for users and items. To get short descriptions of these content profile while preserving the essential statistical relationships that are useful for inferring preference, we extract a content topic feature for each user and item by the probabilistic topic modeling method Latent Dirichlet Allocation (LDA) [7].

Step 2: *We sample negative examples*. In OCCF problems, the data are unbalanced with no negative feedback. In this step, we sample negative examples from missing user-item pairs according to the content topic feature dissimilarity. Thus, we will have both positive and negative examples to learn the model in the next step.

Step 3: *We predict users' missing ratings by Content-dissimilarity Weighted MF*. In this step, we extend Matrix Factorization model by assigning each negative example a weight, which stands for the confidence to treat it as negative. This weight is also determined by the user's and item's content topic feature dissimilarity. After this step, the user and item latent factor can be learned to estimate users' missing ratings on items.

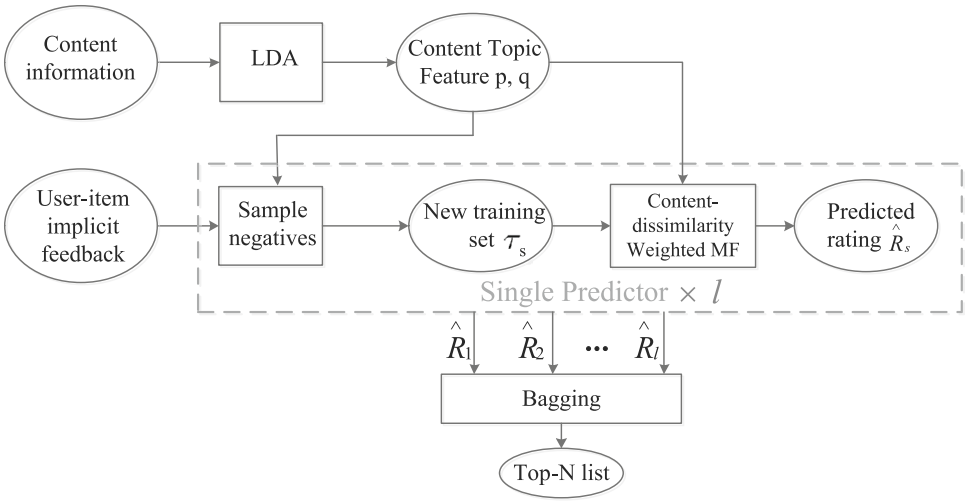

Fig. 1 The flowchart of our algorithm. In the *dotted box*, step 2 and step 3 compose a single predictor. To avoid the biased and unstable predicted results, we repeat it l times, leading to l predictors and get an ensemble of the l predicted results for final recommendation

Step 4: *We get the final recommendation by bagging*. To avoid the biased and unstable predicted results caused by sampling, we will repeat the second and third steps several times and use the bagging technique [9] to construct an ensemble of the results. Finally, we provide a top-N sorted list of items for each user according to the aggregated ratings.

4.2 Content topic feature

In this paper, we propose to exploit the useful content information of social media to identify potential negative examples for OCCF problems.

First of all, we should build the content profile for each user and item. For items in social media websites, we have sufficient information to characterize them. For example, the articles have abstract and text description; the videos have tags and visual feature. For users, the registration information is usually used to profile users in prior works. However, in many social media websites, the registration information is very sparse and limited in reflecting users' characteristics. As we know, user's active actions on social media strongly indicate their preferences; therefore, the user's content profiles could be extracted from the content information of those items she/he liked before, which is easy to obtain. For example, if user u_i has liked $\{v_1, v_2, v_3\}$, we use the content information extracted from $\{v_1, v_2, v_3\}$ to represent user u_i's content profile.

As we know, the content information of social media is usually collections of discrete data. For instance, the text description of articles contains hundreds of words; the visual feature of videos contains hundreds of visual words. Many of these words have similar semantic topic. Since we will refer to the similarity of content profile to determine the potential negative examples, we should make the content feature more semantically concentrated. Thus, we propose to extract *content topic feature* for each user and item, which is the short and semantical description of the content profile, by the probabilistic topic modeling method Latent Dirichlet Allocation (LDA) [7].

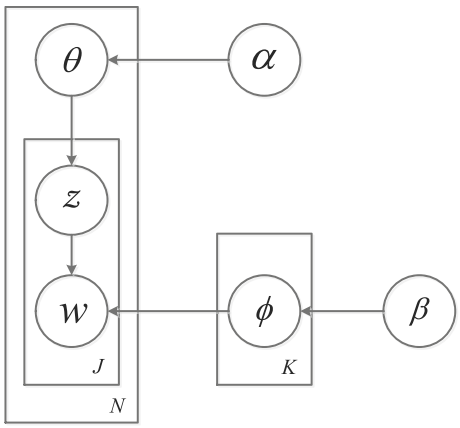

Fig. 2 The graphical model for LDA

Latent Dirichlet Allocation (LDA) is a popular probabilistic topic model to discover a set of "topics" from a large collection of discrete data. It assumes a generative probabilistic model in which documents are represented as random mixtures over latent topics, each topic is characterized by a distribution over words. Assume there are J words in the vocabulary, K latent topics and N documents. Each document j has a topic distribution θ_j, which is a K-dimensional vector, and each topic k has a word distribution ϕ_k, which is a J-dimensional vector. As shown in Fig. 2, the generative process of LDA can be summarized as follows. For each document j in the corpus,

1. Choose topic distribution $\theta_j \sim \text{Dirichlet}(\alpha)$.
2. Choose word distribution $\phi_k \sim \text{Dirichlet}(\beta)$.

3. For each of the word s,
   (a) Choose a topic assignment $z_{j,s} \sim \text{Multinomial}(\theta_j)$.
   (b) Choose a word $w_{j,s} \sim \text{Multinomial}(\beta_{z_{j,s}})$.

This process reveals how the words of each document are assumed to come from a mixture of topics. Given a collection, the posterior distribution (or maximum likelihood estimate) of the topics reveals the K topics that likely generated its documents.

Given a corpus of items, we can use variational EM [7] or Gibbs sampling method [15] to learn the topic-word distribution ϕ and the item-topic distribution θ, where the θ_j is the content topic feature we want to get. Further, for each user, given the user's content information described before, we can use variational inference to situate its content in terms of the topics and get the content topic feature for user. For simplicity, we note the item content topic feature as q and user-content topic feature as p in the rest of the paper. Since the content topic features are interpretable low-dimensional representations of users' and items' content, we measure the content similarity between user u_i and item v_j by:

$$\text{sim}(p_i, q_j) = \frac{p_i^T q_j}{\|p_i\|_2 \times \|q_j\|_2} \tag{3}$$

Note that, if utilizing multiple modalities of content information, for example, the videos have text description and visual feature, we can learn the q and p under each modality as the way described above, and then compute the content similarities under different modalities and integrate them to determine the final similarity between user and item.

4.3 Sampling negative examples

In the OCCF problems, the data are unbalanced with no negative feedback. Traditional methods treat all the missing user-item pairs as negative [6, 19]. On the one hand, there may be many unlabeled positive examples among the missing data. On the other hand, it is still unbalanced with too many negative examples, which makes the model costly to learn in the last step and may bias the results toward negative. In this paper, we sample negative examples from the missing user-item pairs according to the content topic feature dissimilarity. That is, the more dissimilar between user and item content, the higher probability we sample them as negative. Thus, the sampled probability for a missing user-item pair as negative is as follows:

$$P_{ij} \propto 1/\text{sim}(p_i, q_j) \tag{4}$$

The number of our sampled negative examples is comparable with the number of positive examples. Thus, we will have a balanced set of positive and negative examples, denoted as $\mathcal{T}$, to train the model.

4.4 Content-dissimilarity weighted MF

One intuitive way to get the final recommendation is to train the MF model described in Sect. 3.2 with the sampled negative examples and the existing positive examples. However, we can not make sure the sampled negative examples are actually negative. Inspired by the idea of [19, 35] to introduce weight for each example, we assign a weight to each negative example to response the confidence to treat it as negative. Instead of using a global weighting scheme in prior works, a better way is to consider the similarity between the users' and items' content information, that is the more similar they are, more likely the user will like the item, and the less weight we should assign to that negative example. Thus, we have following weight to each negative example:

$$C_{ij} = 1 - \text{sim}(p_i, q_j) \tag{5}$$

For each positive example $C_{ij} = 1$. Therefore, our Content-dissimilarity Weighted MF have the following loss function:

$$Loss(\mathbf{U}, \mathbf{V}) = \sum_{R_{ij} \in \mathcal{T}} C_{ij} \left(R_{ij} - U_i^T V_j\right)^2 + \lambda \left(\|U_i\|_F^2 + \|V_j\|_F^2 \right) \tag{6}$$

where U_i is the latent factor of user u_i, V_j is the latent factor of item v_j. $\mathbf{U} = (U_1, \ldots, U_m) \in \mathrm{R}^{f \times m}$, $\mathbf{V} = (V_1, \ldots, V_n) \in \mathrm{R}^{f \times n}$. $\mathcal{T}$ is the training set containing sampled negative examples and existing positive ones.

The Alternating Least Squares (ALS) method [53] is efficient for solving these kinds of low-rank approximation problems, which rotates between fixing the $\mathbf{U}$ and fixing the $\mathbf{V}$. When all U_i are fixed, the system computes the V_j by solving a least squares problem, and vice versa. Here, we estimate $\mathbf{U}, \mathbf{V}$ by the ALS as follows:

Algorithm 1 ALS for Content-dissimilarity Weighted MF

Require: Training rating set $\mathcal{T}$, content topic feature p and q, latent factor number f
Ensure: latent factor matrices $\mathbf{U}$ and $\mathbf{V}$

Initial $\mathbf{V}$;
if $R_{ij} = 0, \quad \forall R_{ij} \in \mathcal{T}$
  $C_{ij} = 1 - sim(p_i, q_j)$;
else $C_{ij} = 1$;
end if
repeat
  $U_i = (V_{I_i^U} \tilde{C}_{I_i^U} V_{I_i^U}^T + \lambda E)^{-1} V_{I_i^U} \tilde{C}_{I_i^U} R^T(i, I_i^U)$, $\forall 1 \le i \le m$;
  $V_j = (U_{I_j^V} \bar{C}_{I_j^V} U_{I_j^V}^T + \lambda E)^{-1} U_{I_j^V} \bar{C}_{I_j^V} R(I_j^V, j)$, $\forall 1 \le j \le n$;
Until convergence
return $\mathbf{U}$ and $\mathbf{V}$

Fixing $\mathbf{V}$, in order to learn $\mathbf{U}$ which minimize Eq. (6), we take partial derivatives of $Loss(\mathbf{U}, \mathbf{V})$, with respect to each entry of $\mathbf{U}$:

$$\frac{1}{2}\frac{\partial Loss(\mathbf{U},\mathbf{V})}{\partial U_{ri}} = \sum_{j \in I_i^U} C_{ij}\left(U_i^T V_j - R_{ij}\right) V_{rj} + \lambda U_{ri}, \quad \forall 1 \le i \le m, 1 \le r \le f \tag{7}$$

where U_{ri} denotes the rth element of vector U_i, V_{rj} denotes the rth element of vector V_j. I_i^U denotes the set of indices of items that user u_i has rated (no matter negative or positive) in the training set. Then for U_i we have:

$$\begin{aligned}\frac{1}{2}\frac{\partial Loss(\mathbf{U},\mathbf{V})}{\partial U_i} &= \frac{1}{2}\left(\frac{\partial Loss(\mathbf{U},\mathbf{V})}{\partial U_{1i}}; \ldots; \frac{\partial Loss(\mathbf{U},\mathbf{V})}{\partial U_{fi}}\right) \\ &= (V_{I_i^U} \tilde{C}_{I_i^U} V_{I_i^U}^T + \lambda E) U_i - V_{I_i^U} \tilde{C}_{I_i^U} R^T\left(i, I_i^U\right)\end{aligned} \tag{8}$$

where E is an $f \times f$ identify matrix, and $R(i, I_i^U)$ is the row vector where columns $j \in I_i^U$ of the i-th row of R are selected. $V_{I_i^U}$ denotes the sub-matrix of V where columns $j \in I_i^U$ are selected. $\tilde{C}_{I_i^U}$ is a $n_{u_i} \times n_{u_i}$ diagonal matrix with entries of $C(i, I_i^U)$ on the diagonal, where n_{u_i} is the number of items rated by u_i.

Let the partial derivative $\frac{1}{2}\frac{\partial Loss(U,V)}{\partial U_i} = 0$, we get

$$U_i = \left(V_{I_i^U} \tilde{C}_{I_i^U} V_{I_i^U}^T + \lambda E\right)^{-1} V_{I_i^U} \tilde{C}_{I_i^U} R^T\left(i, I_i^U\right) \tag{9}$$

Similarly, by fixing U and solving $\frac{1}{2}\frac{\partial Loss(U,V)}{\partial V_j} = 0$, we have:

$$V_j = \left(U_{I_j^V} \bar{C}_{I_j^V} U_{I_j^V}^T + \lambda E\right)^{-1} U_{I_j^V} \bar{C}_{I_j^V} R\left(I_j^V, j\right) \tag{10}$$

where I_j^V denotes the set of indices of users who have rated v_j (no matter negative or positive) in the training set. $R(I_j^V, j)$ is the column vector where rows $i \in I_j^V$ of the jth column of R are selected. $U_{I_j^V}$ denotes the sub-matrix of U where columns $i \in I_j^V$ are selected. $\bar{C}_{I_j^V}$ is a $n_{v_j} \times n_{v_j}$ diagonal matrix with entries of $C(I_j^V, j)$ on the diagonal, where n_{v_j} is the number of users who have rated v_j.

Based on Eqs. (9) and (10), the algorithm for estimating the latent factors $\mathbf{U}$ and $\mathbf{V}$ is described in Algorithm 1. After achieving them, we can predict the missing data by $\hat{R}_{ij} = U_i^T V_j$, where high score infers higher preference.

4.5 Bagging

To provide a sorted list of items $L^i = (L_1, L_2, \ldots, L_N)$ for each user u_i, which is the aim of recommendation for social media, we can rank the items which have not been rated by u_i according to the ratings $\hat{R}_{ij}$ predicted in above steps. However, since the negative example in the training set is sampled, the predicted rating $\hat{R}_{ij}$ learned from them may be biased and unstable. A practical solution to the problem is to construct an ensemble. Here, we use the bagging technique [9]; the process is shown in Algorithm 2.

Algorithm 2 Bagging for Final Recommendation

Require: observed positive examples R_{ij}, content topic feature p and q, number of single predictor l, number of recommendation results N
Ensure: sorted list of top-N items $L^i = (L_1, L_2, \ldots, L_N), 1 \le i \le m$

for $s = 1 : l$ **do**
  Generate a new training set $\mathcal{T}_s$ by negative example sampling (section 4.3);
  Learn latent factors $\mathbf{U}$ and $\mathbf{V}$ by Algorithm 1;
  Predict rating matrix by $\hat{\mathbf{R}}_\mathbf{s} = \mathbf{U}^T\mathbf{V}$;
end for
Aggregate the predicted rating matrix $\hat{\mathbf{R}} = \frac{1}{l}\sum_{s=1}^{l} \hat{\mathbf{R}}_\mathbf{s}$;
for $i = 1 : m$
  sorted items which have not been rated by u_i, according to the predicted rating $\hat{R}_{ij}$;
end for
return the list of top-N items $L^i = (L_1, L_2, \ldots, L_N), 1 \le i \le m$

5 Experiments

In this section, we investigate the performance of our method in the one-class recommendation problem of social media, by comparing with other state-of-the-art algorithms on two real-world datasets.

Table 1 Statistics of the datasets

	CiteUlike	YouTube
# of users	5,551	4,083
# of items	16,980	9,013
# of observed ratings	204,986	29,879
Ave ratings per user	37	7
Ave ratings per item	12	3
Max ratings per user	403	80
Min ratings per user	10	1
Rating sparsity	99.78 %	99.92 %

5.1 Datasets

CiteUlike dataset CiteUlike[7] is a scientific article sharing service where users create personal libraries by posting the articles they like on the website. Each article has information such as title, abstract, authors, publications and keywords. Recommending potential enjoyed articles for each user in CiteUlike is typically a one-class problem, where we only know the articles she/he likes. To evaluate our models' recommendation quality, we conduct our experiment on a subset of CiteUlike's "user-post-article" data, for each article we use the title and abstract as the content information. The dataset contains 5,551 users and 16,980 articles with 204,986 observed user-item pairs, where each pair denotes a posting action.

YouTube dataset YouTube[8] is a well-known video-sharing website on which users can upload, view and share videos. Each member of YouTube maintains a list of "favored videos" to indicate explicitly her/his preference on those videos. However, there is no clear evidence for us to know explicitly which videos she/he do not like. Thus, it is also a one-class recommendation problem in YouTube. The dataset we collected consists of 7,008 users and 106,693 videos with 349,965 observed user-video pairs, where each pair denotes a "favor" action. We also collected each video's content information, such as the title, the tag, the textual description and the visual information.

The detailed statistics of the two datasets are showed in Table 1. In both datasets, the high sparsity is rather noticeable in the user-item rating matrixes. Compared with CiteUlike dataset, in Youtube dataset, each user rated fewer items and each item was rated by fewer users, which leads to a much sparser rating matrix.

[7] http://www.citeulike.org/.

[8] http://www.youtube.com/.

5.2 Experimental setups

Since the real recommender systems are normally concerned about personalized ranking of items but not rating prediction to all items, we focus on the task of top-N recommendation [12] to evaluate our model's recommendation quality, which provide each user a sorted list of top-N items and evaluate the performance by measuring how well the recommended items hit the users' interests. To achieve this, in the experiments, the observed dataset for each user is randomly divided into twofolds, we use 80 % of the known positive examples for training and the rest 20 % for testing. That is, we train the model on the training set and then give our predictions of the top-N items for each user to see whether the recommended items hit the items in the testing set (i.e., whether the items in the testing set are recommended by the method). The better they hits the testing set, the better recommendation performance the model achieves. The random selection for the training and testing set was carried out 20 times independently, and we report the average results.

For the evaluation protocol, we follow the evaluation mechanism described in [12]. As the total number of items is huge in the datasets, while the number of positive items for each user in the testing set is much fewer, it is prohibitive to take all the items as the candidates and generate a total ordering of the whole item set for each user. Our testing methodology for top-N measure is as follows: For each user u_i, we randomly sample M items that are not viewed by u_i and mix them with the testing data to construct a probe set. Then, we compute the predicted ratings over the probe set to find the top-N items in the probe set. In our experiments, $M = 2{,}000$.

To analyze how well the recommended item list of each method hit the testing items, we consider three classical measures in our experiments: Precision, Recall and MAP (Mean Average Precision).

For each user u_i, given a ranked list of N items, we denote $prec_{u_i}(N)$ as the precision at ranked position N, $recall_{u_i}(N)$ as the recall at ranked position N. Thus, we have:

$$prec_{u_i}(N) = \frac{\#\text{PositiveHits}_{u_i}(N)}{N} \tag{11}$$

$$recall_{u_i}(N) = \frac{\#\text{PositiveHits}_{u_i}(N)}{\#\text{AllPositives}_{u_i}} \tag{12}$$

where $\#\text{AllPositives}_{u_i}$ denotes the total number of positive items which the user u_i likes in the testing set, and $\#PositiveHits_{u_i}(N)$ denotes the number of positive items contained in the ranked position N. The final Precision and Recall at ranked position N, denoted as $Pre(N)$ and

$Recall(N)$, are computed as the mean of $prec_{u_i}(N)$ and $recall_{u_i}(N)$ over all the users.

Mean average precision (MAP) assesses the overall performance based on precisions at different recall levels. It calculates the mean of average precision (AP) over all users in the testing set, where AP for user u_i is the average of precisions computed at all positions with a preferred item:

$$AP_{u_i} = \frac{\sum_{k=1}^{N} prec_{u_i}(k) \times pref_{u_i}(k)}{\#\text{AllPositives}_{\text{u}_\text{i}}} \quad (13)$$

where $prec_{u_i}(k)$ is the precision at ranked position $k, pref_{u_i}(k)$ is a binary indicator returning 1 if the k-th item is preferred or 0 otherwise. The MAP is the mean of AP_{u_i} over all users.

For the above three metrics (Precision, Recall, MAP), a larger value means the recommended list hits the testing set better, which denote a better performance for the task of top-N recommendation [12, 34, 35].

To extract content topic feature, for CiteUlike, we use the title and abstract of each article as the content information. We remove stop words and use tf-idf to choose the top 8,000 distinct words as the vocabulary. Then, the content information can be presented by the counts of these words. The content topic feature can be extracted as described in Sect. 4.2. For YouTube dataset, we use the text features of each video, such as the title, the tag and the textual description, as well as the visual information. For the text features, we can extract the content topic feature $p^{(a)}$ and $q^{(a)}$ as the same way in CiteUlike. For visual information, we first obtain the key frame for each video and then describe them by the typical 2,000-dimensional bag-of-words representation using SIFT [29]. Then, the content topic feature $p^{(b)}$ and $q^{(b)}$ under the visual modality can be extracted as described in Sect. 4.2. Finally, we integrate the content topic feature under two modalities to determine the content similarity between user and item as: $\alpha \text{sim}(p^{(a)}, q^{(a)}) + (1-\alpha)\text{sim}(p^{(b)}, q^{(b)})$. In the experiment of YouTube dataset, $\alpha = 0.7$.

5.3 Baseline models

To evaluate the performance of our model, we implement the following related works for comparison with our model.

- **AMAN**: The traditional CF Matrix Factorization method which treats all the missing data as negative. The optimization problem is as Eq. (2), with all the positive examples assigned to 1 and all the missing examples assigned to 0.
- **AMAU**: The traditional CF Matrix Factorization method which treats all the missing data as unknown. The optimization problem is as Eq. (3), with only the positive examples assigned to 1 and no 0 ones.
- **LDA**: Here, LDA stands for a Content-based method, where we use the LDA-content topic features (discussed in Sect. 4.2) to represent the users' and items' content profile in traditional Content-based method and compute the similarity between them (as Eq. (3)) to produce the top-N list of items for each user.
- **wAMAN_uniform**: The state-of-the-art OCCF algorithm which is the weighted version of AMAN. It treats all the missing data as negative and give a weight to each of them. The weight here is uniform with value less than 1 [34, 35]. This weighting implements the intuition that the confidence of unknown data being negative is lower than the confidence of positive examples.
- **wAMAN_item**: Here, the weights are not-uniform, and the jth item has its own weight proportional to $m - \sum_i R_{ij}$, which is a item-specific weighting scheme, taking the intuition that if an item is viewed by less users, the missing data for this item is negative with higher probability [34, 35].
- **wAMAN_user**: Here, the weights are not-uniform, and the ith user has her/his own weight proportional to $\sum_j R_{ij}$, which is a user-specific weighting scheme, taking the intuition that if a user has viewed more users, those items she/he has not viewed could be negative with higher probability [34, 35].

5.4 Results and analysis

Performance on CiteUlike Table 2 shows the experimental results on the CiteUlike dataset with three different evaluation metrics: MAP, Recall and Precision, when we return top $N = 20$ articles for each user. Through cross-validation, we select the regularizing parameter $\lambda = 1$ and the dimension of content topic feature $K = 50$. We try different number of latent factors f from 10 to 200 and find that the performance increases with the f increasing. $f = 200$ is chosen in reporting the results.

Table 2 Performance comparison on CiteUlike with $N = 20$

Methods	MAP	Recall	Precision
LDA	0.1193	0.3659	0.1099
AMAU	0.0161	0.0417	0.0185
AMAN	0.1735	0.3943	0.1350
wAMAN_uniform	0.2324	0.4563	0.1358
wAMAN_item	0.2129	0.4347	0.1305
wAMAN_user	0.2114	0.4372	0.1306
Ours	**0.2840**	**0.5399**	**0.1630**

Bold values indicate the best performance in comparison

Table 3 Performance comparison on YouTube with $N = 20$

Methods	MAP	Recall	Precision
LDA	0.0171	0.0510	0.0047
AMAU	0.0061	0.0290	0.0026
AMAN	0.0368	0.0898	0.0094
wAMAN_uniform	0.0530	0.1975	0.0199
wAMAN_item	0.0326	0.0792	0.0082
wAMAN_user	0.0484	0.1481	0.0139
Ours	**0.0823**	**0.2282**	**0.0218**

Bold values indicate the best performance in comparison

As we know, a larger value of the three metrics (MAP, Recall and Precision) means a better performance. From Table 2, we can observe that AMAU returns the worst performance in CiteUlike. Even the Content-based method LDA outperforms AMAU, which shows that the content information helps much in article recommendation. AMAN performs better than AMAU and LDA and shows that the negative examples are necessary for recommendation with implicit data. The three OCCF methods focus on weighting negatives increase the result of AMAN and demonstrate that it is useful to model the negative examples in OCCF. However, the user-oriented and item-oriented weighting schemes do not perform as well as the simple uniform scheme. The reason may be that the CiteUlike data are very sparse with most users posting few articles and most articles appearing in few users' libraries, which leads that the weighting scheme based on the number of ratings for each user or item become very weak. It is obvious that our method outperforms other approaches, with significant improvement of 0.0516 (22.20 %) in MAP compared with the best baseline wAMAN_uniform. This suggests that our method can model the negative examples better with the content topic feature and is efficient to solve the sparse and unbalance problems in OCCF.

Performance on YouTube For YouTube dataset, we select the regularizing parameter $\lambda = 0.4$ and the dimension of content topic feature $K = 50$ by cross-validation. We try different number of latent factors f from 10 to 100 and find that the performance increases with the f increasing before $f = 50$ and there seems little improvement after that. Thus, $f = 50$ is chosen in the experiment. The experimental results on YouTube with number of returned videos $N = 20$ are reported in Table 3. Since the YouTube data are much sparser and each user has watched few videos, it is much more difficult for us to hit the favored videos in the test data; thus, the performance of all the methods is much lower than in CiteUlike. However, we can observe that our method also achieves the best performance over all the baseline methods in Youtube dataset. Compared with state-of-the-art method wAMAN_uniform, our method gets a very significant improvement of 55.28 % in MAP, which shows that the content information is very useful in this kind of sparse datasets and our method can model the negative examples better with the content topic feature. Note that the wAMAN_item method does not achieve better performance than AMAN in YouTube dataset, which shows that the strategy to model negative

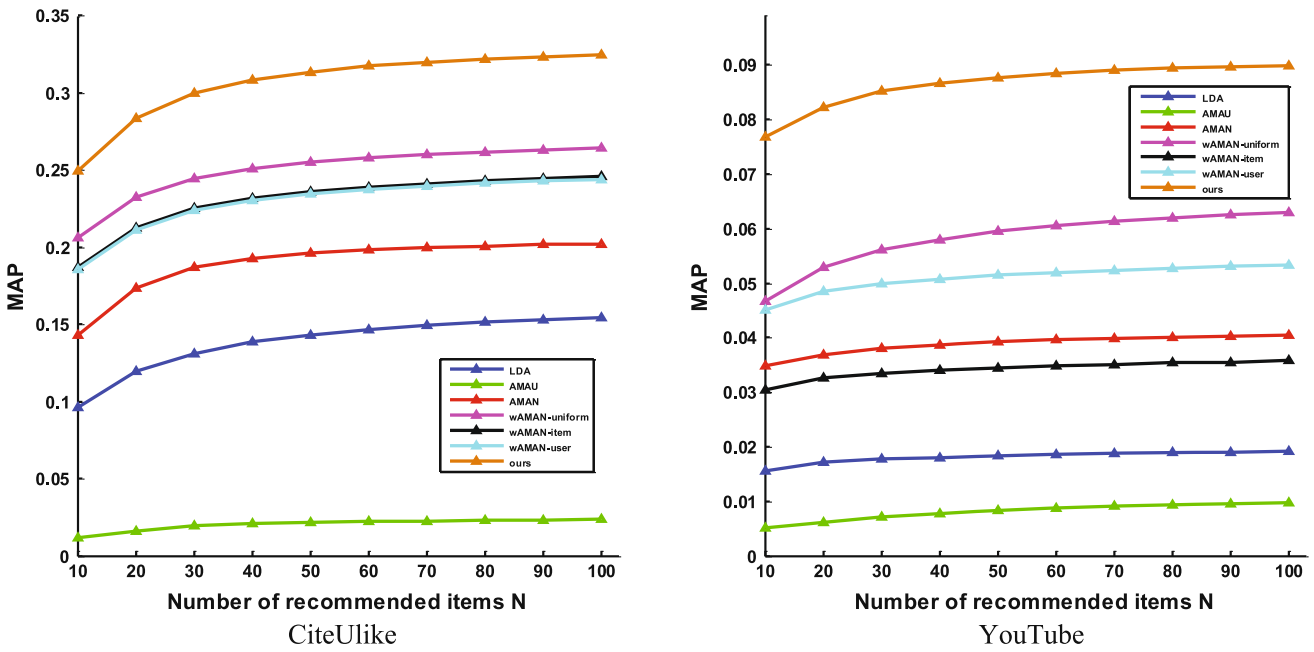

Fig. 3 MAP performance on CiteUlike and YouTube datasets varying different number of recommended items

examples by simply observing the statistical properties of historical feedback is not always suitable for all the datasets.

Performance under different number of recommended items Since our aim was to recommend a top N sorted list of items for each user, different number of recommended items may result in different performance. Figure 3 shows the MAP performance of all the methods with different number of returned items N. From the results, we can observe that on both datasets our method consistently outperforms other approaches varying with different number of recommended items, which demonstrates the high efficiency of our methods.

6 Conclusion

In this paper, we propose a novel framework to deal with the one-class recommendation problems of social media by exploiting the rich content information. Specifically, we get a content topic feature for each user and item to assist distinguishing the potential negative examples from missing data and extend the Matrix Factorization (MF) model by incorporating the content-dissimilarity based weighting scheme. Experiments on real-world data from CiteUlike and YouTube show that the proposed method outperforms state-of-the-art algorithms, which suggests that the our method is effective to overcome the sparsity and unbalance of one-class problem in social media recommendation.

References

1. Abel, F., Gao, Q., Houben, G.J., Tao, K.: Analyzing user modeling on twitter for personalized news recommendations. User Model. Adapt. Pers. **6787**, 1–12 (2011)
2. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. **17**(6), 734–749 (2005)
3. Balabanovic, M., Shoham, Y.: Fab: content-based, collaborative recommendation. Commun. ACM **40**(3), 66–72 (1997)
4. Bao, B.K., Min, W., Lu, K., Xu, C.: Social event detection with robust high-order co-clustering. In: Proceedings of the 3rd ACM Conference on International Conference on Multimedia Retrieval, pp. 135–142 (2013)
5. Bao, B.K., Min, W., Sang, J., Xu, C.: Multimedia news digger on emerging topics from social streams. In: Proceedings of the 20th ACM International Conference on Multimedia, pp. 1357–1358 (2012)
6. Billsus, D., Pazzani, M.J.: Learning collaborative information filters. ICML **98**, 46–54 (1998)
7. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
8. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, pp. 43–52 (1998)
9. Breiman, L.: Bagging predictors. Mach. Learn. **24**(2), 123–140 (1996)
10. Burke, R.: Hybrid web recommender systems. In: The Adaptive Web, pp. 377–408. Springer, Heidelberg (2007)
11. Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., Sartin, M.: Combining content-based and collaborative filters in an online newspaper. In: Proceedings of ACM SIGIR Workshop on Recommender Systems, vol. 60 (1999)
12. Cremonesi, P., Koren, Y., Turrin, R.: Performance of recommender algorithms on top-n recommendation tasks. In: Proceedings of the Fourth ACM Conference on Recommender Systems, pp. 39–46 (2010)
13. Deng, Z., Sang, J., Xu, C.: Personalized video recommendation based on cross-platform user modeling. In: IEEE International Conference on Multimedia and Expo(ICME), pp. 1–6 (2013)
14. Deshpande, M., Karypis, G.: Item-based top-n recommendation algorithms. ACM Trans. Inf. Syst. **22**, 143–177 (2004)
15. Griffiths, T.L., Steyvers, M.: Finding scientific topics. Proc. Natl. Acad. Sci. USA. **101**(Suppl 1), 5228–5235 (2004)
16. Harvey, M., Carman, M.J., Ruthven, I., Crestani, F.: Bayesian latent variable models for collaborative item rating prediction. In: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, pp. 699–708 (2011)
17. Herlocker, J.L., Konstan, J.A., Borchers, A., Riedl, J.: An algorithmic framework for performing collaborative filtering. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 230–237 (1999)
18. Hofmann, T.: Latent semantic models for collaborative filtering. ACM Trans. Inf. Syst. **22**(1), 89–115 (2004)
19. Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: ICDM, pp. 263–272 (2008)
20. Jin, R., Chai, J.Y., Si, L.: An automatic weighting scheme for collaborative filtering. In: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 337–344 (2004)
21. Jin, R., Si, L., Zhai, C.: Preference-based graphic models for collaborative filtering. In: Proceedings of the Nineteenth Conference on Uncertainty in, Artificial Intelligence, UAI'03, pp. 329–336 (2003)
22. Kohrs, A., Merialdo, B.: Clustering for collaborative filtering applications. In. Computational Intelligence for Modelling, Control & Automation, IOS (1999)
23. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 426–434 (2008)
24. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009)
25. Lang, K.: Newsweeder: learning to filter netnews. In: Proceedings of the 12th International Machine Learning Conference (ML95 (1995)
26. Li, X., Snoek, C.G., Worring, M.: Learning social tag relevance by neighbor voting. IEEE Trans. Multimedia **11**(7), 1310–1322 (2009)
27. Linden, G., Smith, B., York, J.: Amazon. com recommendations: item-to-item collaborative filtering. IEEE Internet Comput. **7**, 76–80 (2003)
28. Liu, D., Hua, X.S., Yang, L., Wang, M., Zhang, H.J.: Tag ranking. In: Proceedings of the 18th International Conference on World Wide Web, pp. 351–360 (2009)
29. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. IJCV **60**(2), 91–110 (2004)
30. Mei, T., Yang, B., Hua, X.S., Yang, L., Yang, S.Q., Li, S.: Videoreach: an online video recommendation system. In: ACM SIGIR, pp. 767–768 (2007)

31. Melville, P., Mooney, R.J., Nagarajan, R.: Content-boosted collaborative filtering for improved recommendations. In: AAAI, pp. 187–192 (2002)
32. Mooney, R.J., Roy, L.: Content-based book recommending using learning for text categorization. In: Proceedings of the Fifth ACM Conference on Digital Libraries, pp. 195–204 (2000)
33. OConnor, M., Herlocker, J.: Clustering items for collaborative filtering. In: Proceedings of the ACM SIGIR Workshop on Recommender Systems (1999)
34. Pan, R., Scholz, M.: Mind the gaps: weighting the unknown in large-scale one-class collaborative filtering. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 667–676 (2009)
35. Pan, R., Zhou, Y., Cao, B., Liu, N.N., Lukose, R., Scholz, M., Yang, Q.: One-class collaborative filtering. In: ICDM, pp. 502–511 (2008)
36. Pazzani, M., Billsus, D.: Learning and revising user profiles: the identification of interesting web sites. Mach. Learn. **27**(3), 313–331 (1997)
37. Pazzani, M.J.: A framework for collaborative, content-based and demographic filtering. Artif. Intel. Rev. **13**(5–6), 393–408 (1999)
38. Rennie, J.D.M., Srebro, N.: Fast maximum margin matrix factorization for collaborative prediction. In: Proceedings of the 22nd International Conference on Machine Learning, pp. 713–719 (2005)
39. Salakhutdinov, R., Mnih, A.: Probabilistic matrix factorization. In: Advances in Neural Information Processing Systems (2008)
40. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th International Conference on World Wide Web, pp. 285–295 (2001)
41. Schein, A.I., Popescul, A., Ungar, L.H., Pennock, D.M.: Methods and metrics for cold-start recommendations. In: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 253–260 (2002)
42. Shen, J., Wang, M., Yan, S., Cui, P.: Multimedia recommendation: technology and techniques. In: Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1131–1131. ACM (2013)
43. Si, L., Jin, R.: Flexible mixture model for collaborative filtering. In: ICML, pp. 704–711 (2003)
44. Sindhwani, V., Bucak, S., Hu, J., Mojsilovic, A.: A family of non-negative matrix factorizations for one-class collaborative filtering problems. In: Proceedings of the ACM Recommender Systems Conference, RecSys (2009)
45. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. Adv. Artif. Intel. **2009**, 1–19 (2009)
46. Teh, Y.W., Jordan, M.I., Beal, M.J., Blei, D.M.: Hierarchical dirichlet processes. J. Am. Stat. Assoc. **101**(476), 1566–1581 (2006)
47. Tiemann, M., Pauws, S.: Towards ensemble learning for hybrid music recommendation. In: Proceedings of the 2007 ACM Conference on Recommender Systems, pp. 177–178 (2007)
48. Wang, C., Blei, D.M.: Collaborative topic modeling for recommending scientific articles. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 448–456. ACM (2011)
49. Wang, Z., Sun, L., Zhu, W., Yang, S., Li, H., Wu, D.: Joint social and content recommendation for user-generated videos in online social network. IEEE Trans. Multimedia **15**(3), 698–709 (2013)
50. Wang, Z., Zhu, W., Chen, X., Sun, L., Liu, J., Chen, M., Cui, P., Yang, S.: Propagation-based social-aware multimedia content distribution. ACM Trans. Multimedia Comput. Commun. Appl. **9**(1s), 52 (2013)
51. Wang, Z., Zhu, W., Cui, P., Sun, L., Yang, S.: Social media recommendation. In: Social Media Retrieval, pp. 23–42. Springer, Berlin (2013)
52. Zhang, Y., Koren, J.: Efficient bayesian hierarchical user modeling for recommendation system. In: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 47–54 (2007)
53. Zhou, Y., Wilkinson, D., Schreiber, R., Pan, R.: Large-scale parallel collaborative filtering for the netflix prize, pp. 337–348 (2008)