Optimize Recommendation System with Topic Modeling and Clustering

Conference Paper · November 2017

DOI: 10.1109/ICEBE.2017.13

Qianqiao Liang
Zhejiang University

Optimize Recommendation System with Topic Modeling and Clustering

Qianqiao Liang, Xiaolin Zheng*, Menghan Wang
College of Computer Science
Zhejiang University
{liangqq, xlzheng, wangmengh}@zju.edu.cn

Haodong Chen
Hithink RoyalFlush Information Network Co., Ltd
chenhaodong@myhexin.com

Pin Lu
Zhejiang Fangyuan Test Co., Ltd
hzpinlu@163.com

***Abstract*—With the rapid development of e-commerce, recommender systems have been widely studied. Many recommendation algorithms utilize ratings and reviews information. However, as the number of users and items grows, these algorithms face the problems of sparsity and scalability. Those problems make it hard to extract useful information from a highly sparse rating matrix and to apply a trained model to larger datasets. In this paper, we aim at solving the sparsity and scalability problems using rating and review information. Three possible solutions for sparsity and scalability problems are concluded and a novel recommendation model called TCR which combines those three ideas are proposed. Experiments on real-world datasets show that our proposed method has better performance on top-N recommendation and has better scalability compared to the state-of-the-art models.**

keywords*—*Topic Modeling, Clustering, Recommender System

I. Introduction

A recommender system is a kind of information filtering system that applies some machine learning algorithms to predict the rating or preference from a user to an item. Recommender systems are widely used in e-commerce such as Taobao and Amazon. It not only helps customers identify commodities they may like, but also increases companies' profits. Therefore, recommendation algorithms have been developed over the years, which can be roughly categorized into two groups: collaborative filtering based [1] and content based [2] methods.

Collaborative filtering(CF) based methods use the past records for recommendation, such as users' purchase history or users' ratings. Users' ratings are usually represented by a user-item rating matrix, where each element is one user's rating towards an item. CF based methods find similar users or items using the rating matrix and recommend within the neighbourhood. It's a widely used method. However, with the increase of the number of users and items, CF based methods face the problems of sparsity and scalability. The problem of sparsity makes it hard to extract useful information from a highly sparse rating matrix while the problem of scalability make it hard to maintain the performance of a trained model when datasets become larger.

*Corresponding Author.
Tel.: 0086-0571-87951245
Email address: xlzheng@zju.edu.cn

To solve the sparsity problem, three main aspects can be considered. Firstly, we can integrate CF based mtehods with content based methods using additional information such as review texts, graph information [3] and point of interest [4]. Specifically, abundant review texts in e-commerce platform make it available to use topic model in recommendation. zTopic model [7], which is a hot topic in the field of natural language processing, is an important way to improve the accuracy of recommendation. For example, Hong and Davison [8] and Yan et al. [9] apply topic model to short texts of Twitter and conduct in-depth analysis. Secondly, we can use matrix decomposition to extract latent features, which can alleviate the curse of dimensionality of a sparse rating matrix. For example, since stochastic gradient descent(SGD) is a common matrix factorization algorithm due to its efficiency, Tan et al. [16] combine topic model with SGD to extract latent features from rating matrix. At the same time, Jing [10] uses Laplace distribution for matrix fatorization. Thirdly, we can make dataset denser, which means clustering data into smaller and denser subsets. For example, Yi et al. [5] cast clustering into a matrix completion problem. Yin and Wang [6] cluster texts using an online clustering scheme for initialization.

However, those three algorithms above are still flawed. Topic modeling algorithm in recommendation is complex, which usually requires a lot of computing, resulting in low scalability of the model. Matrix decomposition algorithm can not be applied directly to rating matrix because missing records are represented by zeros in rating matrix. Therefore, the precision of matrix decomposition is challenged. Besides, clustering algorithms also depend on latent features which are extracted using matrix decomposition. In addition, existing clustering algorithms always assume that one person belongs to one cluster, while in real scenarios one may belong to multiple clusters due to various interests.

In this paper, we try to alleviate the disadvantages mentioned above and integrate the ideas of those three solutions. We propose a hybrid recommendation model called Topic Clustering Recommendation (TCR), which can slove the problem of sparsity and scalability with lower complexity. We first present how those three algorithms can be better used in recommendation systems and then present our own model which integrates those three optimizations. Experiments show that our hybrid model significantly outperforms three state-of-the-art models. Specifically, main contributions of this paper can be summarized as follows:

- We summarize three solutions to sparsity and scalability problems, point out the shortcomings of those existing solutions, and propose a simpler and more efficient approach, which is a new approach to combine collaborative filtering and content-based recommendations.
- We conduct experiments on five real-world datasets to evaluate the effectiveness of our hybrid model. Experimental results show that our hybrid model outperforms novel clustering and topic modeling methods in terms of top-N recommendation (recommendation of N highest scored commodities) and model training time.

The rest of this paper is organized as follows: In Section II, we normalize the problem settings and nomenclature used throughout this paper. Then we outline related works. In Section III, we describe our proposed method, that is, TCR for recommendation. In section IV, we evaluate the performance of our TCR model on five real-world datasets and compare our TCR model with three state-of-the-art clustering model and topic model. Finally, we make some conclusions and propose the future work.

II. Preliminaries

In this section, we start with normalizing the problem setting and nomenclature used throughout this paper. Then we introduce relevant topic model, matrix decomposition model and clustering model, which will provide theoretical basis for our proposed model in this paper.

A. Problem Definition

We consider the recommendation task as a rating prediciton problem and a ranking prediciton problem. In a standard recommendation setting, we have m users, n items, and an extremely sparse rating matrix $\mathbf{R} \in \mathbb{R}^{m\times n}$. Each entry $\mathbf{R}_{ij}$ of $\mathbf{R}$ corresponds to user i's rating on item j. If $\mathbf{R}_{ij} \neq 0$, it means the rating of user i on item j is observed, otherwise unobserved. Additionally, we use term frequency – inverse document frequency(TF-IDF) weighted bag-of-words model to construct the users' and items' review descriptor w. Each item's reviews can be represented as a vector. Let $w_j = \{w_{j1}, w_{j2}, ..., w_{j|N|}\}$ be the word vector for the j-th item v_j, where N is the index set of words and $w_{jd}(d \in N)$ represents the number of appearances of word d in v_j's review texts. Similarly, Let $w_i = \{w_{i1}, w_{i2}, ..., w_{i|N|}\}$ be the word vector for user u_i.

Let x_i, $y_j \in \mathbb{R}^{1\times K}$ be user i's and item j's latent feature vectors respectively extracted from both ratings and reviews, where K is the dimensionality of latent space. And the corresponding matrix forms of latent feature vectors for all users and items are $X = x_{1:m}$ and $Y = y_{1:n}$, respectively. Also, we denote $H^T = \begin{bmatrix} X^T & Y^T \end{bmatrix}$. Then we use these latent feature vectors for clustering and let $P \in \mathbb{R}^{(m+n)\times C}$ records cluster distrubution for each user and item, where C is the number of classes.

Given the sparse rating matrix $\mathbf{R}$ and those review texts, our goal is to learn users' latent feature vectors X and items' latent feature vectors Y, use those vectors for clustering and hence to predict the missing ratings in $\mathbf{R}$. Finally, we sort the predicted scores and recommend N highest scored items for users, which is called top-N recommendation.

B. Topic Model

Topic model is a probabilistic model that utilizes textual information to exploit latent features. The most commonly used method for topic model is latent dirichlet allocation(LDA) [18]. Hong and Davison [8] and Yan et al. [9] apply LDA to short texts of Twitter and conduct in-depth analysis.

LDA assumes that each document is a mixture of a few topics, and each word of each document is subordinate to a topic. It also assumes that prior distribution of θ and ϕ is Dirichlet distribution with parameters α and β respectively, where $\theta \in \mathbb{R}^{(m+n)\times T}$ and $\phi \in \mathbb{R}^{T\times|N|}$ are document-topic distribution the topic-word distribution respectively with T topics. w and z are subject to multinomial distributions, where $z \in \mathbf{R}^{(m+n)\times|N|}$ records the topic of each word in each document.

Among those parameters above, only w is what we can observe. The goal of LDA model is to find other hidden parameters or hidden distributions under a given observable w. In LDA model, the generation process of $m+n$ documents can be described as follows:

1. Choose $\theta_i \sim Dirichlet(\alpha)$ where $i \in \{1, 2, ..., m+n\}$.
2. Choose $\phi_t \sim Dirichlet(\beta)$ where $t \in \{1, 2, ..., T\}$.
3. For each of the word positions i, j, where $i \in \{1, 2, ..., m\}$ and $j \in \{1, 2, ..., n\}$.
    1) Choose a topic $z_{ij} \sim Multinomial(\theta_i)$
    2) Choose a word $w_{ij} \sim Multinomial(\phi_{z_{ij}})$

Through the comparison of generated documents with real documents, we optimize those LDA parameters, and finally get θ as latent feature vectors extracted from review texts.

C. Matrix Decomposition Method

Many commercial recommender systems are used to evaluate very large product sets. User-item rating matrix will thus be extremely sparse and the dimension of the matrix is high. So we can extract latent features in a much lower dimension. For example, Recht et al. [11] use lock free approach. Later, they also proposed DSGD [12] partitioning the rating matrix into independent blocks. On the other hand, [13] [14] propose to use coordinate descent instead of SGD to optimize the loss function. However, traditional matrix decomposition algorithm can not be applied directly to the rating matrix because missing records are represented by zeros in rating matrix. Hence, we will introduce a matrix decomposition method which ignores those zeros in rating matrix. This method is proposed in a novel model called MCoC [15].

MCoC model assumes that latent vectors of user i (x_i) and item j (y_j) should be very close if they have a high rating score. And for those missing value pairs, there is no restriction to their latent vectors. So it adopts the following loss function to model the user-item relationships:

$$\epsilon(X,Y) = \sum_{i=1}^{m}\sum_{j=1}^{n}(||\frac{x_i}{\sqrt{D_{ii}^{row}}} - \frac{y_j}{\sqrt{D_{jj}^{col}}}||^2 R_{ij}) \quad (1)$$

where $D^{row} \in \mathbb{R}^{m\times m}$ is the diagonal degree matrix of users with $D_{ii}^{row} = \sum_{j=1}^{n} R_{ij}$ and $D^{col} \in \mathbb{R}^{n\times n}$ is the diagonal degree matrix of items with $D_{jj}^{col} = \sum_{i=1}^{m} R_{ij}$.

By some simple linear algebra derivations, we can get:

$$\begin{aligned}\epsilon(X,Y) &= \sum_{i=1}^{m}||x_i||^2 + \sum_{j=1}^{n}||y_j||^2 - \sum_{i=1}^{m}\sum_{j=1}^{n}\frac{2x_i^T y_j R_{ij}}{\sqrt{D_{ii}^{row}D_{jj}^{col}}} \\ &= Tr(X^TX + Y^TY - 2X^TSY) \\ &= Tr(\begin{bmatrix}X^T & Y^T\end{bmatrix}\begin{bmatrix}I_m & -S \\ -S^T & I_n\end{bmatrix}\begin{bmatrix}X \\ Y\end{bmatrix}) \\ &= Tr(H^TMH).\end{aligned} \quad (2)$$

where

$$S = (D^{row})^{-\frac{1}{2}}R(D^{col})^{-\frac{1}{2}}. \quad (3)$$

$$M = \begin{bmatrix}I_n & -S \\ -S^T & I_m\end{bmatrix}. \quad (4)$$

M is a semi-definite matrix. So similar to the spectral clustering model [20] and the bipartite graph model [21], the optimal solution H^* that minimizes the loss function (1) is given by the solution of eigenvalue problem $MH = \lambda H$. So $H^* = [h1, ..., hr]$, where $h_1, ..., h_r$ are the smallest eigenvectors of matrix M sorted by their corresponding eigenvalues. In this way, matrix factorization is irrelevant to zeros in matrix.

D. Fuzzy C-Means Clustering Model

Clustering is a task of grouping a set of objects that objects in the same cluster are more similar to each other than to those in other clusters. Co-clustering [19] and semi-supervised clustering like [5] have been developed but one big limitation of them is that, each user or item can be clustered into one single cluster only. In fact, users are allowed to have a variety of hobbies, that is, they can belong to multiple clusters. Therefore, we use fuzzy c-means model(FCM) as our clustering model. FCM's procedure can be described as follows:

1. Select the number of clusters C, randomly initialize the center of each class, and set the convergence condition.
2. Repeat the following steps until the convergence condition is reached:
    1) Calculate the center of each class

$$c_j = [\sum_{i=1}^{m+n} P_{ij}^2 h_i]/[\sum_{i=1}^{m+n} P_{ij}^2] \quad (5)$$

    2) For each user or item, calculate its distribution on each class

$$P_{ij} = (d(h_i, c_j)^{-2})/[\sum_{k=1}^{C}(d(h_i, c_k))^{-2}] \quad (6)$$

where d is a distance function which can be predefined and h_i is the $i\text{-}th$ row of H. In this paper, we use Euclidean distance for calculation.

III. Topic Clustering Recommendation

In this section, we describe our proposed method TCR for recommendation. Our primary goal is to alleviate the disadvantages of those three algorithms and integrate the ideas of those three solutions discussed in the first section. TCR is such an integrated model that combines the idea of topic modeling and clustering. Then we use TCR to improve the performance of sparsity and scalability in recommender systems. There are three main questions we need to answer:

1. How to represent review information? Can review information be represented in the same form as rating matrix? Rating information is represented by a matrix, and topic model is applied to the review texts to obtain the latent feature vectors of users and items. How should we express information from the reviews that correspond to the rating information?
2. How does the rating information and review texts combine to extract latent feature vectors? Because the existing algorithms to combine both of them are complex, we have to propose a simpler synthesis model to solve the problem of scalability.
3. How to recommend after clustering to solve the sparsity problem?

Our algorithm is to answer these three questions. We use a review similar matrix to represent correlations between users and items which are extracted from review texts. So that the review information and the rating information can be expressed in the same form. Similar to the MCoC model, we use the properties of the semi-definite matrix to simplify the computational complexity and improve the scalability of the model in the process of extracting the latent eigenvectors. Finally, in the recommended process, we smooth the ratings among all clusters to predict unobserved data and make recommendation.

A. Topic Modeling

A review text is used to extract the topic of an article. Since we gather all the reviews of one user or item, which are represented by w_i and w_j, those review texts can be used to represent the topic of users and items. We will prove this assumption in our experiments. We need to randomly select different themes for each word in observed texts using Gibbs sampling method until the maximum likelihood of the words is optimized. Finally, as we discussed in the previous section, θ is what we need.

Next, we will denote the information reflected in the review text as a similarity matrix. If user i has a score record for item j, then pearson correlation coefficient of θ_i and θ_j is calculated as the similarity of the two vectors. Review similarity matrix $Sim \in \mathbb{R}^{m\times n}$ which records the correlation coefficient of users to items is then constructed. The formula for the Pearson correlation coefficient is as follows:

$$Sim_{i,j} = \frac{(\theta_i - \overline{\theta_i})(\theta_j - \overline{\theta_j})}{||\theta_i - \overline{\theta_i}||_2||\theta_j - \overline{\theta_j}||_2} \tag{7}$$

where $\overline{\theta_i}$ and $\overline{\theta_j}$ are the mean values of θ_i and θ_j. Obviously, the value range of $Sim_{i,j}$ is [-1, 1]. And because of the subsequent matrix factorization, we need to ensure that the elements of the review similarity matrix are nonnegative. So that by equally scaling, we make the value range of elements in review similarity matrix the same as the rating matrix, that is [0, 5].

B. Latent Frature Extraction

In this step, we need to get latent feature vectors of users and items based on the rating matrix and review similar matrix. Obviously, the higher the user's rating on the item, and the higher the similarity reflected by the user's reviews on the item, the more likely the user and the item are in the same subclass, and the closer their latent feature vectors are. According to this assumption, a loss function is proposed:

$$\begin{aligned}\epsilon(X,Y) = &\sum_{i=1}^{m}\sum_{j=1}^{n}(||\frac{x_i}{\sqrt{D_{ii}^{row}}} - \frac{y_j}{\sqrt{D_{jj}^{col}}}||^2 R_{ij}) \\ &+ \sum_{i=1}^{m}\sum_{j=1}^{n}(||\frac{x_i}{\sqrt{F_{ii}^{row}}} - \frac{y_j}{\sqrt{F_{jj}^{col}}}||^2 Sim_{ij})\end{aligned} \tag{8}$$

where F is similarly defined as D: $F^{row} \in \mathbb{R}^{m\times m}$ is the diagonal degree matrix of users with $F_{ii}^{row} = \sum_{j=1}^{n} Sim_{ij}$ and $F^{col} \in \mathbb{R}^{n\times n}$ is the diagonal degree matrix of items with $F_{jj}^{col} = \sum_{i=1}^{m} Sim_{ij}$. Also, the loss function can be derived as:

$$\begin{aligned}\epsilon(X,Y) &= \sum_{i=1}^{m}||x_i||^2 + \sum_{j=1}^{n}||y_j||^2 - \sum_{i=1}^{m}\sum_{j=1}^{n}\frac{2x_i^T y_j R_{ij}}{\sqrt{D_{ii}^{row}D_{jj}^{col}}} \\ &\quad - \sum_{i=1}^{m}\sum_{j=1}^{n}\frac{2x_i^T y_j Sim_{ij}}{\sqrt{F_{ii}^{row}F_{jj}^{col}}} \\ &= Tr(2X^TX + 2Y^TY - 2X^TSY - 2X^TVY) \\ &= Tr(\begin{bmatrix}X^T & Y^T\end{bmatrix}\begin{bmatrix}2I_m & -S-V \\ -S^T-V^T & 2I_n\end{bmatrix}\begin{bmatrix}X \\ Y\end{bmatrix}) \\ &= Tr(H^TM^*H).\end{aligned} \tag{9}$$

where

$$V = (F^{row})^{-\frac{1}{2}} Sim (F^{col})^{-\frac{1}{2}} \tag{10}$$

$$M^* = \begin{bmatrix}2I_n & -S-V \\ -S^T-V^T & 2I_m\end{bmatrix}. \tag{11}$$

Similar as the discussion in MCoC model, the optimal solution H^* that minimizes the loss function (8) is given by the solution of eigenvalue problem $M^*H = \lambda H$. So $H^* = [h1, ..., hr]$, where $h_1, ..., h_r$ are the smallest eigenvectors of matrix M^* sorted by their corresponding eigenvalues.

C. FCM Clustering

After getting the users' and items' latnet feature vectors, FCM clustering is required. Formula (5) and (6) are repeated until convergence and the final P is what we needed. P represents the distribution of users and items on C categories. But in fact, each user's interest is limited while the type of item is also limited. So each user or item will only belong to k categories among all those C classes. Therefore, it is necessary to set the smallest $(C-k)$ elements of each row in P to zero and then normalize each row of P.

D. Smooth Probability Recommendation

Rating prediction can be made after matrix P is computed above. In order to solve the problem of sparsity and scalablility, we use the method of probability model to synthesize users' ratings and items' ratings in C categories with their clusters' distribution. We calculate the probability of a score that a user will give to an item, and finally take the expected rating score.

We denote the probability of user i in cluster c to be:

$$p(c|i) = P_{i,c} \tag{12}$$

We assume that in each category, an item's ratings match the Gaussian distribution, which means that the probability of an item being scored r in one class is:

$$p(r;\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma}exp[-\frac{(r-\mu)^2}{2\sigma^2}] \tag{13}$$

Where μ and σ represent the average and variance of the item in each class, and the formulas are as follows:

$$\mu_{j,c} = \frac{\sum_{i\in I_c} r_{i,j}p(c|i)}{\sum_{i\in I_c} p(c|i)} \tag{14}$$

$$\sigma_{j,c} = \frac{\sum_{i\in I_c}(r_{i,j} - \mu_{j,c})^2 p(c|i)}{\sum_{i\in I_c} p(c|i)} \tag{15}$$

Where I_c is item set of the c-th cluster. Combing (13) – (15), we have:

$$p(r|i,j) = \sum_c p(c|i)p(r|c,j) = \sum_c p(c|i)p(r;\mu_{j,c},\sigma_{j,c}) \tag{16}$$

$$E[r|i,j] = \int rp(r|i,j)dr = \sum_c p(c|i)\mu_{j,c} \tag{17}$$

where $E[r|i,j]$ is the expected rating, that is, rating prediction of user i to item j. And then a list of ranked items is generated for each user based on these predicted ratings.

E. Algorithm Overview

Here we make a brief summary of our framework for top-N recommendation. In practice, we do following steps in $\mathbf{Algorithm 1}$.

```
Algorithm 1 Topic Clustering Recommendation
INPUT: Rating matrix $R$. Review texts in the form of $w$
OUTPUT: A list of ranked items for each user.
1: Apply LDA topic model on all $w$ : randomly select different themes for each word in observed text using the Gibbs sampling method until the maximum likelihood of the words is optimized. Get $\theta$ for each user and item.
2: Compute the correlation coeficient of $\theta_i$ and $\theta_j$ if user $i$ has comment on item $j$, and compute review similarity matrix $Sim$ according to (7).
3: Extract latent feature vectors $H(X$ and $Y)$ according to loss function (8).
4: Conduct FCM clustering using $H$. Repeat formula (5) and (6) until convergence and get $P$.
5: Predict the unobserved ratings using formular (17). And then generated a list of ranked items for each user.
```

We generate review similar matrix and rating matrix according to review and rating infomation. Using those two matrices, we extract latent feature vector for each user and item. Finally, clustering and recommendation are conducted based on those latent feature vectors. Fig.1 shows the graphical model for TCR.

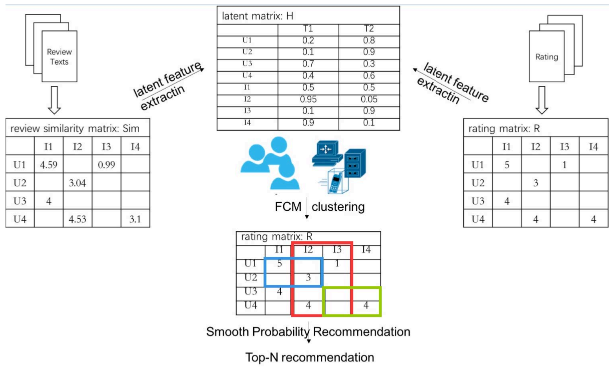


Fig. 1. The graphical model for the TCR.

IV. Experiment

In this section, we evaluate the performance of our TCR model with five real-world datasets from Amazon [22], and compare our TCR model with three state-of-the-art recommendation model.

A. Dataset

The experiment data for this paper comes from the purchase records from Amazon. In Amazon data, there are 24 different product categories of data records. We select 5 of them for model verification. For each record, we use the user ID, item ID, rating, and comment text.

We firstly clean the data: only to retain users who purchase more than 20 items and their corresponding items. The information of those selected datasets is presented as follows:

TABLE I
Dataset Description

Dataset	#user	#item	#rating	sparseness
Office Produce	545	1897	16819	0.9837
Health & Personal Care	1969	13007	75104	0.9971
Grocery & Gourmet Food	1238	7334	45810	0.9950
Tools & Home	530	6874	16904	0.9954
Toys & Games	877	8587	31319	0.9958

As for review texts, we remove high frequency and meaningless stop words. After that, we use TF-IDF method to calculate the word frequency. In data processing, we find that Amazon's ratings are relatively high with the average value of 4. So in the actual recommendation, we believe that only records with score of 5 are our successful recommendations. So the validation set and test set are further cleaned, retaining only the data with a score of 5.

In parameter selection step, we randomly divide datasets into training set and validation set according to the ratio 4: 1. In model comparison step, we randomly divide datasets into training set and test set according to the ratio 4: 1.

B. Experiment Settings

1) Baseline: We compare our proposed method TCR with the following methods:

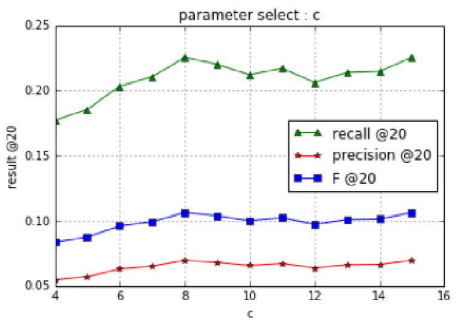
- RBLT [16]. Rating boost latent topic is a novel model which maps eigenvectors extracted from reviews and ratings into the same dimension for recommendaion.
- MCoC [15]. Multi co-clusteting is a widely used model which conduct clustering based on rating matrix.
- MC [17] Matrix completion algorithm fills the user-item matrix based on a low-rank assumption and simultaneously keep the original information.

2) Experimental Environment: All the experiments are implemented using Python and are conducted on a four-core processor with processing frequency of 2.60GHz and memory of 4GB. Each result is the average result after 10 times of experiments.

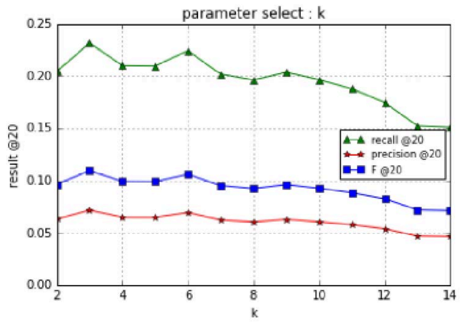
3) Evaluation Metrics: We use recommendation accuracy, which is commonly used to evaluate the performance of a recommendation system. Recommendation accuracy consists of precision, recall and F. Precision@20, recall@20, and F@20 in the experiment indicate the recommendation accuracy with 20 items recommended in the candidate set. The formulas are as follows:

$$precision = \frac{\sum_{u \in U} hits_u}{N \cdot |U|}. \tag{18}$$

$$recall = \frac{\sum_{u \in U} hits_u}{h \cdot |U|}. \tag{19}$$



(a) Recall@20 at different C with k fixed to 3



(b) Recall@20 at different k with C fixed to 15

Fig. 2. Parameter selection with different C and k

$$F = \frac{2 \cdot precision \cdot recall}{precision + recall}. \tag{20}$$

where $hits_u$ indicates the number of items that users actually like, that is, successful recommendation on average in the top-N item recommendation. h represents the average number of potential products in the validation set or test set per user. $|U|$ indicates the size of the user set.

4) Settings: For the construction of w, we assume that the maximum number of corpus is 3000. We filter words whose appearance times are less than 3 or the appeарence frequancy are higher than 0.9. For LDA model, we choose 20 topics for each user and item. We set the hyperparameters as $\alpha = 0.05, \beta = 0.01$, which is the common values of most applicaitons.

C. Experiment Results and Analysis

We run state-of-the-art recommendation models and compare their top-N recommendation performance with TCR. After that, we make an empirical analysis to the results, as well as the sparsity and scalability. Finally, we have a short discussion on future work.

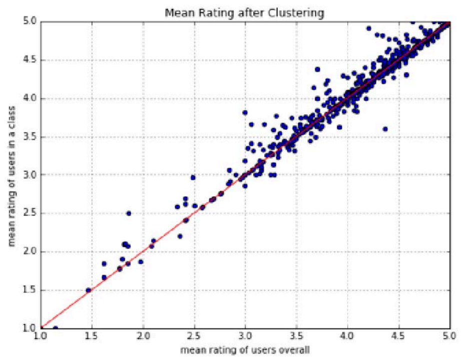


Fig. 3. Average score comparison before clustering and after clustering

1) Parameter Selection: In model comparison, we assume that there are many categories of goods, but a person's preference is limited and will only be interested in some of the categories. For example, there are a few categories in Baby dataset such as baby's bottle, baby's clothing, movers diapers and toys. But a customer may interested in two of them. Therefore, we think that C can be large, and k to be small. The experiment is to find the best proportion of C and k for subsequent experiments. We fix k to 3 and observe recall@20 under different C values. Similarly, we fix C to 15 and observe recall@20 under different k values.

Fig.2(a) shows that when k is fixed to 3, with C being larger, the recommended result is firstly rising and then being stable. This is because when a user's preference category is limited to k, if the value of C is too close to k, the distinction between users is not obvious, which is basically the same as no clustering. On the other hand, if C is too large, it will increase the computational complexity. So from fig.2(a), we choose the turning point of recommended result from rising to stable, concluded that when k = 3, C is considered to be 8. Fig.2(b) shows that when C is fixed at 15, with k being larger, the recommended result is first stable and then decreases. When C is large, it imeans that there are many subclasses of such goods, so the number of subclasses that users may be interested in will increase. Similarly, from Fig.2(b), we select the inflection point k = 6. After several experiments, we conclude that the relationship between C and K is roughly $k = \lceil \log_2 C \rceil$. In this paper, because Amazon dataset has divided a general category of data, we believe that the number of sub-categories under each data set will not be too much. we finally choose $C = 8$ and $k = 3$ for futher experiments.

After selecting C, k, we can visualize the effect of clustering. The specific evaluation index is whether the user's or item's score in each category after clustering is higher than the average score in the whole case.

Results show that 70.1% of users' average ratings have improved in their largest probability clusters compared to their average ratings overall while 72.14% of items' average ratings have improved in their largest probability clusters compared to their average ratings overall. Fig.3 can visually show the results, where the data points(users) on the left to the line have their performance improved. This proves that users are more satisfied with items in those clusters, which proves a feasible solution of sparseness.

2) Model Comparison: In this experiment, we first experiment on those baselines to verify the recall@20 on 5 datasets, and then use the results of the parameter selection experiment to perform TCR model on 5 datasets.

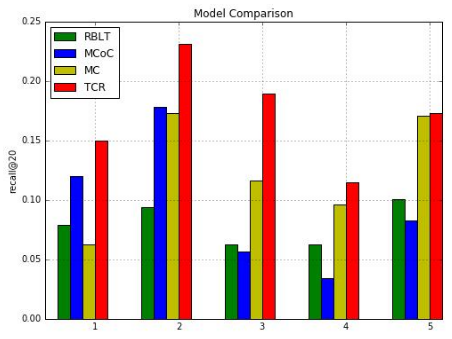


Fig. 4. User's average score comparison before clustering and after clustering

Fig.4 shows that TCR performs better than MCoC, RBLT and MC models among all 5 datasets. This also validates the ideas presented in this paper: TCR model that combines topic model and clustering model can improve the recommendation results compared to a single topic model or a single cluster model. TCR also outperforms MC model, whose latent feature extraction methos does not consider missing values seperatly in rating matrix.

3) New User Test: Scalability means that when the data size becomes larger, the algorithm can still maintain its performance. To solve the problem of scalability, we can consider training a model offline and observe the recommendation results of new users or new products (users or items that are not used in model training). This type of test is also called out-of-matrix test. In this paper, out-of-matrix experiment refers to recommendation for new users and observation of their recall. At the same time, reducing the complexity of the algorithm is also another way to solve the problem of scalability, so we will also record the corresponding offline training time and online predicting time. Finally, we will compare the total runtime with increasing size of datasets.

Since the traditional collaborative filtering method can not conduct out-of-matrix test, the experiment only shows the out-of-matrix recommendation of the TCR model and RBLT. In out-of-matrix test, we conduct offline training and online recommendations. In offline training, Items' latent feature vectors are computed using RBLT or TCR algorithms. In online recommendations, new users' latent factors are then represented by an average across items they rated, weighted by degree of ratings. In this lab, we only select two of these datasets for experiments: the grocery and gourmet food dataset and the office product dataset. For each dataset, we select 50 users, who will only appear in the test set.

As we can see in table II, when a new user comes, the new user's recall@20 is also ideal in TCR. For grocery and gourmet food dataset, TCR's out-of-matrix recommendation recall reduces by 8.67% compared with the offline recommendation recall, while the RBLT recommendation recall is reduced by 60.5.61%. For office product dataset, TCR's out-of-matrix recommendation recall reduces by 17.66% compared with the offline recommendation result, while the RBLT recommendation result is reduced by 79.15%. RBLT's online recommendation performance reduction is basically intolerable and invalid.

On the other hand, model's training time can reflect the model complexity. From this point of view, TCR's training time on grocery and gourmet food dataset is 70.65% shorter than RBLT's, and also 63.96% shorter in the office product dataset. It can be seen that complexity of TCR model is lower and the scalability of it is better than the traditional topic model. Although the out-of-matrix recommendation time of TCR is a little longer than RBLT because online recommendation for TCR requires the comparison of distance from each user to each cluster center, we consider it to be worthy for a higher recommendation accuracy.

Finally, we want to see whether the algorithm can still control its runtime when the data size becomes larger. According to table I , the grocery and gourmet food dataset is 63.28% bigger the office product dataset. But the total runtime is only 52.68% longer in TCR model. Therefore, runtime can be well control within linear complexity in our proposed method.

4) Example Analysis: This experiment is designed to observe some specific results recommended by the TCR. According items' ID in datasets, we can find the corresponding specific goods from the Amazon website. We observe whether the user's preferences reflected by the topic words is consistent with those recommended results. Table III shows a randomly selected user in Amazon's digital music dataset [22]. We present the user's most correlated topics and his recommended goods computed by TCR. The last column of the table indicates whether the recommendation is successful (whether it is in the test set).

Obviously, user 1 likes to buy songs on CD with type of hip-hop, rap or other fast-paced songs. Among the recommended top 10 items, 9 are such songs and 4 are successfully recommended. In addition, from the output of the top 10 topic, we can see that there are some emotional words such as ”like”, ”nice” etc. with their position being front. This inspires us to remove emotional words or make use of them in the future work. Moreover, topic words such as ”track”, ”beat” which describe item properties are also frequent. This inspires us to make use of these property-describe words in the future work.

V. Conclusion

In this paper, we address the problem of sparsity and scalability of the existing models. We conclude three possible solutions to solve those problems: topic modeling, matrix decomposition and clustering. Then we present a hybrid model called TCR which integrates those three ideas and reduce the model complexity by a simpler latent factor extration technique. Finally, experiments on 5 datasets prove the effectiveness of those three solutions and the comparison to

TABLE II
OUT-OF-MATRIX TEST

	TCR				RBLT			
	Offline Training		Online Prediction		Offline Training		Online Prediction	
	time	recall@20	time	recall@20	time	recall@20	time	recall@20
Grocery and Gourmet	600.00s	18.92%	82.44s	17.28%	2043.93s	4.28%	4.90s	1.69%
Office Product	310.60s	14.95%	12.33s	12.31%	861.86s	5.90%	2.31s	1.23%

TABLE III
EXAMPLE USER

	User 1	Recommended?
Top 3 topics	1.like,track,cd,just,song,tracks,single,know,don,game,time,better,best,people,songs,make	
	2.good,beat,songs ,featuring,great,classic,nice,track,hook,production,best,rap,tight,solo	
	3.rap,hip,hop,production,beats,classic,great,track,tracks,albums,lyrics,beat,song,west,coast	
Top 10 items	1.The Infamous Explicit Lyrics – an album with rap and beat	√
	2.All Eyez on Me Explicit Lyrics – a hip-hop crowns CD	
	3.Liquid Swords Explicit Lyrics – a CD with rapic-fire flow and battle rythmes	
	4.ATLiens Explicit Lyrics – a CD with sound of Atlanta rap	
	5.The Fix Explicit Lyrics – a brilliant hip-hop album with the soul of a blue record	
	6.2Pac Greatest Hits – an collection showcasting the passionate genius of the late rapper	√
	7.Me Against the World – a classic	√
	8.The Don Killuminati – a collection of the singers song	
	9.Reasonable Doubt – popular demand, classic debut album	√
	10.Southen play a list icadillac muzil – a CD that changed hip-hop in the south for years	

baselines indicates that TCR can solve the problem of sparsity and scalability.

VI. ACKNOWLEDGEMENTS

This work was supported in part by the National Natural Science Foundation of China (Nos.61379034 and U1509221), the National Key Technology R&D Program (Nos. 2014BAH28F05, 2015BAH07F01 and 2015BAH17F02), the Zhejiang Province Science and Technology Program (No.2017C03044), and the Guangdong Province Science and Technology Program (No. 2014B040401005).

REFERENCES

[1] X. Su and T. M. Khoshgoftaar, "A Survey of Collaborative Filtering Techniques", *Advances in Artificial Intelligence.*, 2009, Vol.2009, pp.1-19.
[2] G. Adomavicius and A. Tuzhilin, "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions", *IEEE transactions on knowledge and data engineering.*, 2005, Vol.17, No.6, pp.734-749.
[3] N. Rao, H.F. Yu and P. Ravikumar et al., "Collaborative Filtering with Graph Information: Consistency and Scalable Methods",*Advances in Neural Information Processing Systems.* , 2015, pp.2107–2115.
[4] K. Hideaki, I. Tomoharu and F. Yasuhiro et al., "Read the Silence: Well-timed Recommendation via Admixture Marked Point Processes", *Proceedings of the Thirty-First Conference on Artificial Intelligence.*, AAAI. 2017, pp.132-139.
[5] J. Yi., L. Zhang and R. Jin et al., "Semi-supervised Clustering by Input Pattern Assisted Pairwise Similarity Matrix Completion",*Proceedings of the 30th International Conference on Machine Learning.*, ICML. 2013, pp.1400-1408.
[6] J. Yin. and J. Wang, "A Text Clustering Algorithm Using an Online Clustering Scheme for Initialization",*Proceedings of the 22nd International Conference on Knowledge Discovery and Data Mining*, SIGKDD. 2016, pp.1995-2004.
[7] B. Liu, and L. Zhang, "A Survey of Opinion Mining and Sentiment Analysis", *Mining text data.*, Springer US, 2012, pp.415-463.
[8] L. Hong and B. D. Davison, "Empirical Study of Topic Modeling in Twitter",*Proceedings of the first workshop on social media analytics*, 2010, pp.80-88.
[9] X. Yan, J. Guo, and Y. Lan et al., "A Biterm Topic Model for Short Texts", *Proceedings of the 22nd international conference on World Wide Web*, WWW. 2013, pp.1445-1456.
[10] L. Jing, P. Wang, and P. Yang, "Sparse Probabilistic Matrix Factorization by Laplace Distribution for Collaborative Filtering", *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, IJCAI. 2015, pp.1771-1777.
[11] B. Recht, C. Re and S. J. Wright et al. Hogwild, "A Lock-Free Approach to Parallelizing Stochastic Gradient Descent", *Advances in Neural Information Processing Systems.*, NIPS. 2011, pp.693-701.
[12] B. Recht and C. Re, "Parallel Stochastic Gradient Algorithms for Large-Scale Matrix Completion", *Mathematical Programming Computation.*, 2013, Vol.5, No.2, pp.201-226.
[13] H.F. Yu, C.J. Hsieh and S. Si et al., "Parallel Matrix Factorization for Recommender Systems", *Knowledge and Information Systems.*, 2016, Vol.41, No.3, pp.179-185.
[14] H. F. Yu, C. J. Hsieh and S. Si et al., "Scalable Coordinate Descent Approaches to Parallel Matrix Factorization for Recommender Systems", *International Conference on Data Mining*, ICDM. 2015, pp.765-774.
[15] B. Xu, J. Bu and C. Chen et al., "An Exploration of Improving Collaborative Recommender Systems via User-Item Subgroups", *Proceedings of the 21st international conference on World Wide Web.*, WWW. 2012, pp.21-30.
[16] Y. Tan, M. Zhang and Y. Liu et al., "Rating-Boosted Latent Topics: Understanding Users and Items with Ratings and Reviews", *Proceedings of the 25th International Joint Conference on Artificial Intelligence.*, IJCAI. 2016, pp.2640-2646.
[17] Z. Kang., and C. Peng. and Q. Cheng, "Top-N Recommender System via Matrix Completion", *Proceedings of the Thirtieth Conference on Artificial Intelligence.*, AAAI. 2016, pp.179-185.
[18] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation", *Journal of machine Learning research.*, 2003, Vol.3, pp.993-1022.
[19] T. George and S. Merugu, "A Scalable Collaborative Filtering Framework Based on Co-Clustering" *Proceedings of the 5th International Conference on Data Mining.*, ICDM. 2005, Vol.4.
[20] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On Spectral Clustering: Analysis and an Algorithm", *Advances in Neural Information Processing Systems.*, NIPS. 2001, pp.849-856.
[21] I. S. Dhillon, "Co-Clustering Documents and Words using Bipartite Spectral Graph Partitioning", *Proceedings of the seventh international conference on Knowledge discovery and data mining.*, SIGKDD. 2001, pp.269-274.
[22] Julian, M.(2014). *Amazon product data*. University of California, San Diego. Available: from http://jmcauley.ucsd.edu/data/amazon/.